Wedderburn rank reduction and Krylov subspace method for tensor approximation. Part 1: Tucker case

S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov

Institute of Numerical Mathematics, Russian Academy of Sciences,
Russia, 119333 Moscow, Gubkina 8
[sergei.goreinov,ivan.oseledets,dmitry.savostyanov]@gmail.com

October 19, 2010

Abstract. New algorithms are proposed for the Tucker approximation of a 3-tensor, that access it using only the tensor-by-vector-by-vector multiplication subroutine. In the matrix case, Krylov methods are methods of choice to approximate the dominant column and row subspaces of a sparse or structured matrix given through the matrix-by-vector multiplication subroutine. Using the Wedderburn rank reduction formula, we propose an algorithm of matrix approximation that computes Krylov subspaces and allows generalization to the tensor case. Several variants of proposed tensor algorithms differ by pivoting strategies, overall cost and quality of approximation. By convincing numerical experiments we show that the proposed methods are faster and more accurate than the minimal Krylov recursion, proposed recently by Eldén and Savas.

Keywords: Multidimensional arrays, sparse tensors, structured tensors, Tucker approximation, Krylov subspace methods, Wedderburn rank reduction, fast compression
AMS classification: 15A23, 15A69, 65F99

1 Introduction

In this paper we focus on algorithms for the low-rank approximation of large three-dimensional arrays (tensors), that play increasingly important role in many applications.

Throughout the paper, a *tensor* $\mathbf{A} = [a_{ijk}]$ means an array with three indices. The indices are also called *modes* or *axes*. The number of allowed values for a mode index is called the *mode size*. In numerical work with tensors of large mode sizes it is crucial to look for the data sparse structures. Most common are the following.

The *canonical decomposition* [18, 3, 17] (or *canonical approximation*, if the right-hand side does not give $\mathbf{A}$ exactly) of a tensor $\mathbf{A} = [a_{ijk}]$ reads

$$\mathbf{A} = \sum_{s=1}^{R} u_s \otimes v_s \otimes w_s, \qquad a_{ijk} = \sum_{s=1}^{R} u_{is} v_{js} w_{ks}. \qquad (\mathrm{C})$$

The minimal possible number of summands is called the *tensor rank* or the *canonical rank* of a given tensor $\mathbf{A}$. However, the canonical decomposition/approximation of a tensor with minimal value of R is an ill-posed and computationally unstable problem [8], and among several practical algorithms none is known to be absolutely robust.

The (truncated) *Tucker decomposition/approximation* [35] of $\mathbf{A}$ reads

$$\mathbf{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W, \qquad a_{ijk} = \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{s=1}^{r_3} g_{pqs} u_{ip} v_{jq} w_{ks}. \qquad (\mathrm{T})$$

The quantities r_1, r_2, r_3 are referred to as the *Tucker ranks* or the *mode ranks*, the $r_1 \times r_2 \times r_3$ tensor $\mathbf{G} = [g_{pqs}]$ is called the *Tucker core*. In d dimensions, the memory to store the $r \times r \times \ldots \times r$ core is r^d, that is usually beyond affordable for large d and modest r (so-called *curse of dimensionality*). In three dimensions for $r \sim 100$ the storage for the core is small and the Tucker decomposition can be used efficiently.

Here and below, the symbol $\times_l$ designates the multiplication of a tensor by a matrix along the l-th mode. For example, $\mathbf{B} = \mathbf{A} \times_2 M$ means summation on 2nd index $b_{ijk} = \sum_{j'} m_{jj'} a_{ij'k}$. The notation for tensor operations is not yet standard (for current state of art see review [21]), we use the one proposed by de Lathauwer in the article on *multilinear SVD* [6] (or higher-order SVD, HOSVD). In [6] the Tucker format arises with additional constraints: orthogonality of factors, that is also assumed in our paper, and all-orthogonality of the core, that we will relax. For a tensor given as a full array of elements, the multilinear SVD provides a quasi-optimal Tucker approximation which further can be refined by different iterative methods such that Tucker-ALS [22, 7], Newton-Grassmann [10, 19], etc. It reduces to the SVD for three *unfolding matrices* (see [6] and (13) later in this paper) and costs $\mathcal{O}(n^4)$ operations for $n_1 \times n_2 \times n_3$ tensor.[$]
It is clearly too much for large n, and we should look for alternatives.

In the matrix case two classes of fast methods are popular: cross algorithms and Krylov-based approaches. Cross methods compute rank-r approximation by interpolating the $n_1 \times n_2$ matrix on a cleverly chosen set of *crosses*, proposed for instance in [36, 15]. That requires $\mathcal{O}(nr)$ evaluations of matrix elements and can be used for matrices implicitly given by a subroutine that evaluates any prescribed element. However, the verification of the approximation requires several heuristics. If the matrix is structured (sparse, Toeplitz or Hankel, low-rank, sum and/or product of above, etc.) and a

---

[$]We always assume $n_1 = n_2 = n_3 = n$ and $r_1 = r_2 = r_3 = r$ in complexity estimates

fast matrix-by-vector product is available, then the Krylov-type methods are methods of choice with established convergence and complexity estimates.

The generalization of the cross method to 3-tensors (Cross3D, [25]) requires the careful algorithmic implementation and several tweaks to make it really efficient. Generally, the Cross3D interpolates a tensor on $\mathcal{O}(nr + r^3)$ elements and uses $\mathcal{O}(nr^2 + r^4)$ additional operations. However, the accuracy check also requires heuristics.

Recently, Eldén and Savas [30] proposed two generalizations of Krylov methods to tensors: minimal and maximal Krylov recursion. The minimal recursion requires only $3r$ tensor-by-vector-by-vector multiplications to compute basis sets U, V, W for the Tucker rank-(r, r, r) approximation (T), but sometimes it converges slowly and even does not converge (for example, for tensors with sufficiently different mode ranks). The maximal recursion is convergent, but needs definitely unaffordable number of tensor operations. The goal of this paper is to propose algorithms that use tensor only through the tensor-by-vector-by-vector multiplication subroutine, have asymptotical complexity equal to the one of minimal Krylov recursion whereas have better convergence properties. We also try to keep a link with the existing theory of matrix approximation and with ideas used in [25] for the cross approximation of tensors.

This paper is organized as follows. In the section 2 we propose the optimization of minimal Krylov recursion, inspired by the idea of maximization of the orthogonal component of new vectors. In the section 3 we recall the Wedderburn rank reduction formula that gives a nice framework for many matrix factorizations [5]. Then we propose a variant of a matrix decomposition that is similar to the Gaussian elimination with pivoting but uses the matrix through the matrix-by-vector products. In the section 4 we generalize this method to the tensor case. The Wedderburn elimination process gives some freedom in a selection of the vectors to eliminate, and we use it proposing several 'pivoting' strategies, that lead to different complexity estimates and convergence properties. Some pivoting strategies turn the proposed method into the minimal Krylov recursion and the optimized minimal Krylov recursion. In the section 5 we apply the proposed family of algorithms for the approximation of different structured tensors and give numerical comparison with previous methods.

The *tensor-by-vector-by-vector multiplication*, shortly the *tenvec* operation, can be defined via the tensor-by-matrix products. For example, tenvec of $n_1 \times n_2 \times n_3$ tensor $\mathbf{A} = [a_{ijk}]$ at modes $2, 3$ reads

$$u = \mathbf{A} \times_2 v^\mathsf{T} \times_3 w^\mathsf{T}, \qquad u_i = \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} a_{ijk} v_j w_k$$

where v, w are vectors of length n_2 and n_3 respectively and the result is a vector u of length n_1. This is the solely tensor operation in algorithms in this paper and we propose extremely simple notation

$$\mathbf{A} \times_2 v^\mathsf{T} \times_3 w^\mathsf{T} \overset{\text{def}}{=} \mathbf{A}vw, \qquad \mathbf{A} \times_3 w^\mathsf{T} \times_1 u^\mathsf{T} \overset{\text{def}}{=} \mathbf{A}wu, \qquad \mathbf{A} \times_1 u^\mathsf{T} \times_2 v^\mathsf{T} \overset{\text{def}}{=} \mathbf{A}uv,$$

relaxing the information on contraction modes, that is always obvious from the symbolic

notation and mode sizes of vectors. This simple notation (that generalizes the Tdd notation [2]) reveals the analogy of tensor algorithms with the matrix case.

We consider the approximation of matrices and tensors in the Frobenius norm

$$\|\mathbf{A}\|_F^2 \stackrel{\text{def}}{=} \langle \mathbf{A}, \mathbf{A} \rangle, \qquad \langle \mathbf{A}, \mathbf{B} \rangle \stackrel{\text{def}}{=} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \sum_{k=1}^{n_3} a_{ijk} b_{ijk}.$$

For theoretical estimates we also use the spectral norm of tensor (cf. [9])

$$\|\mathbf{A}\|_2 \stackrel{\text{def}}{=} \max_{\|x\|=\|y\|=\|z\|=1} \mathbf{A} \times_1 x^T \times_2 y^T \times_3 z^T = \max_{\|x\|=\|y\|=\|z\|=1} \langle \mathbf{A}, x \otimes y \otimes z \rangle,$$

induced by the standard vector norm $\|x\|^2 \stackrel{\text{def}}{=} \|x\|_2^2 = (x, x) = \sum_{i=1}^n |x_i|^2$.

It is worth to mention that discussed algorithms first aim to construct the orthogonal bases U, V and W of the dominant mode subspaces, and the core $\mathbf{G}$ of the Tucker approximation can be computed afterwards. For the fixed U, V, W the most accurate approximation in Frobenius norm is obtained with the core

$$\mathbf{G} = \mathbf{A} \times_1 U^T \times_2 V^T \times_3 W^T. \tag{1}$$

In general, (1) can be computed using r^2 tenvecs, but faster implementations are available for certain structured tensors and can essentially improve the complexity. We can also be satisfied with the sub-optimal but fast formula for $\mathbf{G}$, for example the one based on interpolation on a maximum-volume set of indices [14] as proposed in [26]. If the computation of (1) is not a necessary part of the discussed algorithm, this cost is not included in the total complexity.

2 Minimal Krylov recursion and its optimization

The generalization of the Krylov subspace method to the problem of tensor approximation was first proposed by Eldén and Savas in [30]. Two variants are discussed, namely the minimal and the maximum Krylov recursion.

The minimal Krylov recursion (MKR, see Alg. 1) requires 3 tenvecs on each iteration, therefore, basis sets $U = U_r, V = V_r, W = W_r$ are computed in $3r$ tenvecs. However, their 'quality', i.e. the accuracy of the approximation $\tilde{\mathbf{A}} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ with the optimal core (1) may be low, and for some cases $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F$ will not reduce at all, even for large ranks.

As an example, consider an $n \times n \times n$ tensor $\mathbf{A}$ with only two non-zero slices[$]

$$\mathbf{A}(:,:,1) = A_1, \qquad \mathbf{A}(:,:,2) = A_2.$$

Obviously, if $A_1 \neq A_2$, the mode-3 subspace of $\mathbf{A}$ consists of two vectors $W = [e_1 \ e_2]$. Starting the MKR Alg. 1 from some u_1, v_1, we accumulate $W_2 = [e_1 e_2]$ in two steps.

[$]Here and after we use the MATLAB-style notation, with ":" denoting all possible index values

---

Algorithm 1: [30] Minimal Krylov recursion for tensor approximation (MKR)

Input: Tenvec subroutine for tensor $\mathbf{A}$
Output: Mode subspaces U, V, W for the Tucker approximation
Initialization: Unit vectors u_1, v_1
   $w_1 := \mathbf{A}u_1v_1/\|\mathbf{A}u_1v_1\|, \quad U_1 = [u_1], V_1 = [v_1], W_1 = [w_1]$
   for $k = 1, 2, \ldots$ **do**
     $u := \mathbf{A}v_kw_k; \; u' := (I - U_kU_k^\mathsf{T})u; \; u_{k+1} := u'/\|u'\|; \; U_{k+1} := [U_k \; u_{k+1}]$
     $v := \mathbf{A}w_ku_{k+1}; \; v' := (I - V_kV_k^\mathsf{T})v; \; v_{k+1} := v'/\|v'\|; \; V_{k+1} := [V_k \; v_{k+1}]$
     $w := \mathbf{A}u_{k+1}v_{k+1}; \; w' := (I - W_kW_k^\mathsf{T})w; \; w_{k+1} := w'/\|w'\|; \; W_{k+1} := [W_k \; w_{k+1}]$
   end for

Then all the computed vectors w have zero component w' orthogonal to W_2. This situation is referred to as *breakdown*, since we cannot continue the process.

In [30] Eldén and Savas propose to fix breakdowns by taking an orthogonal vector to the subspace W. However with $w_3 := e_3$ we come to zero vector $u := \mathbf{A}v_3w_3$ on the next iteration and can not continue the process. Another possible workaround, proposed in [31] is to use the last basis vector on all the subsequent iterations, setting $w_3 := w_2, w_4 := w_2$ and so on. This should be done when the subspace of the mode-3 vectors of $\mathbf{A}$ equals to span W_k. However, it is difficult to check this fact using only tenvec operations and it is not clear how to do this in the non-exact case which is always true in the machine precision arithmetic. In the numerical examples (see section 5.2) we show that the poor convergence and even the stagnation can occur also for 'more practical' situations.

From this discussion we derive the idea of optimization of the Alg. 1. Consider the step for u and generalize it as follows.

$$u := \mathbf{A}(V_k\hat{v}_k)(W_k\hat{w}_k); \quad u' := (I - U_kU_k^\mathsf{T})u; \quad u_{k+1} := u'/\|u'\|; \quad U_{k+1} := [U_k \; u_{k+1}].$$

New direction u is generated by the tenvec of $\mathbf{A}$ with *some* vector from span V and *some* vector from span W. In MKR we always take $\hat{v}_k = \hat{w}_k = e_k$, but there is no background theory behind this choice and it seems to be not optimal in practice. To improve the approximation properties of U_k, we could choose $\hat{v}_k$ and $\hat{w}_k$ to maximize the norm of the orthogonal component u'. Since

$$u' = (I - U_kU_k^\mathsf{T})u = \left(\mathbf{A} \times_1 (I - U_kU_k^\mathsf{T}) \times_2 V_k^\mathsf{T} \times_3 W_k^\mathsf{T}\right) \times_2 \hat{v}_k^\mathsf{T} \times_3 \hat{w}_k^\mathsf{T},$$

we are to find

$$\hat{v}_k, \hat{w}_k = \arg \max_{\|\hat{v}\|=1, \|\hat{w}\|=1} \|\mathbf{B}\hat{v}\hat{w}\| \qquad \text{for} \quad \mathbf{B} = \mathbf{A} \times_1 (I - U_kU_k^\mathsf{T}) \times_2 V_k^\mathsf{T} \times_3 W_k^\mathsf{T}. \qquad (2)$$

The global maximum is not required. We can satisfy with the sufficiently large u' that can be found by several iterations of the *alternating least squares* method (ALS, see [22] and Alg. 2), for which the local linear convergence is proved in the rank-$(1, 1, 1)$ case [7, 38]. If the ALS converges to the best rank-one approximation, it also solves (2)

Algorithm 2: [22] ALS rank-$(1, 1, 1)$ iteration

Input: Tenvec subroutine for tensor $\mathbf{A}$
Output: Best rank-$(1, 1, 1)$ approximation $\sigma u \otimes v \otimes w$ for tensor
Initialization: Unit vectors v, w

  for $k = 1, \dots, p_{\text{als}}$ **do**
    $u := \mathbf{A}vw; \quad \sigma := \|u\|; \quad u := u/\|u\|$
    $v := \mathbf{A}wu; \quad \sigma := \|v\|; \quad v := v/\|v\|$
    $w := \mathbf{A}uv; \quad \sigma := \|w\|; \quad w := w/\|w\|$
  end for

---

as a dual problem. To see this, consider the approximation $\tilde{\mathbf{B}} = b u \otimes \hat{v} \otimes \hat{w}$ with unit $u, \hat{v}, \hat{w}$ and compute the optimal $1 \times 1 \times 1$ core b by (1)

$$b := \mathbf{B} \times_1 u^\mathsf{T} \times_2 \hat{v}^\mathsf{T} \times_3 \hat{w}^\mathsf{T} = \langle \mathbf{B}, u \otimes \hat{v} \otimes \hat{w} \rangle = (\mathbf{B}\hat{v}\hat{w}, u).$$

The approximation $\tilde{\mathbf{B}}$ that minimizes the approximation error

$$\|\mathbf{B} - \tilde{\mathbf{B}}\|_\mathsf{F}^2 = \|\mathbf{B}\|_\mathsf{F}^2 - 2\langle \mathbf{B}, b u \otimes \hat{v} \otimes \hat{w} \rangle + \|b u \otimes \hat{v} \otimes \hat{w}\|_\mathsf{F}^2 = \|\mathbf{B}\|_\mathsf{F}^2 - |b|^2$$

also maximizes $|b|$ and solves (2), since

$$\max_{\|u\|=\|\hat{v}\|=\|\hat{w}\|=1} |b| = \max_{\|\hat{v}\|=\|\hat{w}\|=1} \max_{\|u\|=1} (\mathbf{B}\hat{v}\hat{w}, u) = \max_{\|\hat{v}\|=\|\hat{w}\|=1} \|\mathbf{B}\hat{v}\hat{w}\|.$$

Each ALS iteration requires 3 tenvecs with $\mathbf{B}$, that can be rendered by 3 tenvecs with $\mathbf{A}$ and $\mathcal{O}(nk)$ operations for the orthogonalization. The optimization of the MKR is summarized in terms of tenvecs operations in the Alg. 3.

If p_{als} ALS iterations are used to solve (2) on Step 6, then Alg. 3 requires $(3+9p_{\text{als}})r$ tenvecs and $\mathcal{O}(nr^2 p_{\text{als}})$ additional operations.

In numerical experiments (see section 5) we show that the optimized MKR Alg. 3 shows the better convergence that the MKR Alg. 1. However, convergence estimates are still missing, even in the exact low-rank case. Is it possible to develop a method that has guaranteed convergence at least in the exact case? We start from the matrix case, using Wedderburn rank-one reduction formula. Then, in section 4, we propose generalization of Wedderburn method to tensor case and reintroduce the MKR and the optimized MKR as different versions of the same Wedderburn process, for which we discuss the convergence and give numerical examples and comparison.

3 Matrix approximation using Wedderburn rank reduction formula

3.1 Preliminaries

Many matrix decomposition algorithms can be represented as a sequence of rank-one *Wedderburn updates* [37]. For a matrix A and vectors x, y of appropriate sizes, such

---

Algorithm 3: Optimized minimal Krylov recursion for tensor approximation

---

Input: Tenvec subroutine for tensor $\mathbf{A}$, tolerance parameter `tol`

Output: Mode subspaces U, V, W for approximation $\mathbf{A} \approx \tilde{\mathbf{A}} = \mathbf{G} \times_1 U \times_2 V \times_3 W$

Initialization: Unit vectors u_1, v_1

1: $w_1 := \mathbf{A}u_1v_1/\|\mathbf{A}u_1v_1\|, \quad U_1 = [u_1], V_1 = [v_1], W_1 = [w_1]$
2: $updU = updV = updW = \mathbf{true}; \quad k = l = m = 1$
3: **while** $updU$ or $updV$ or $updW$ **do**
4:    **if** $updU$ **then** {Proceed with the new vector u if required}
5:       Define $\mathbf{B} = \mathbf{A} \times_1 (I - U_k U_k^\mathsf{T}) \times_2 V_l^\mathsf{T} \times_3 W_m^\mathsf{T}$
6:       Find $\hat{v}_k, \hat{w}_k := \arg\max_{\|\hat{v}\|=1, \|\hat{w}\|=1} \|\mathbf{B}\hat{v}\hat{w}\|$ by p_{als} ALS steps, see Alg. 2
7:       $u := \mathbf{A}(V_l \hat{v}_k)(W_m \hat{w}_k); \quad u' := (I - U_k U_k^\mathsf{T})u$
8:       **if** $\|u'\| < \mathtt{tol}\|u\|$ **then** {Breakdown}
9:         Fix breakdown or set $updU := \mathbf{false}$
10:      **else**
11:        $u_{k+1} := u'/\|u'\|; \quad U_{k+1} := [U_k\ u_{k+1}]; \quad k := k + 1$
12:      **end if**
13:    **end if**
14:    {Proceed with new vector v if required}
15:    {Proceed with new vector w if required}
16: **end while**

---

that $x^\mathsf{T} A y \neq 0$, the matrix

$$B = A - \frac{A y x^\mathsf{T} A}{x^\mathsf{T} A y}. \tag{3}$$

has rank $B = \operatorname{rank} A - 1$. For the rank-$r$ matrix $A_0 = A$ after r updates of form

$$A_k = A_{k-1} - \frac{A_{k-1} y_k x_k^\mathsf{T} A_{k-1}}{x_k^\mathsf{T} A_{k-1} y_k} \tag{4}$$

with $\omega_k \stackrel{\text{def}}{=} x_k^\mathsf{T} A_{k-1} y_k \neq 0$, the matrix A_r becomes zero and the rank-r decomposition of A can be constructed.

In [5] the properties of the Wedderburn sequence (4) are studied in much detail. We recall a list of basic facts about (4) in the compact and 'more matrix' form.

Statement 1. Matrix A_k writes as $A_k = P_k^\mathsf{T} A = A Q_k$ with $P_0 = I$, $Q_0 = I$ and

$$P_k = P_{k-1} - \omega_k^{-1} P_{k-1} x_k y_k^\mathsf{T} A^\mathsf{T} P_{k-1}, \qquad Q_k = Q_{k-1} - \omega_k^{-1} Q_{k-1} y_k x_k^\mathsf{T} A Q_{k-1}. \tag{5}$$

Corollary 1. $P_k x_k = 0$ and $Q_k y_k = 0$.

Statement 2. Matrices P_k, Q_k are projectors, i.e. $P_k^2 = P_k$ and $Q_k^2 = Q_k$.

Obviously, $P_0 = P_0^2 = I$, $Q_0 = Q_0^2 = I$ and from $P_{k-1}^2 = P_{k-1}$ it follows that

$$P_k^2 = P_{k-1}^2 - \omega_k^{-1} P_{k-1}^2 y_k x_k^\mathsf{T} A^\mathsf{T} P_{k-1} - \omega_k^{-1} P_{k-1} x_k y_k^\mathsf{T} A^\mathsf{T} P_{k-1}^2 +$$
$$+ \omega_k^{-2} P_{k-1} x_k \underbrace{y_k^\mathsf{T} A^\mathsf{T} P_{k-1}^2 x_k}_{\omega_k} y_k^\mathsf{T} A^\mathsf{T} P_{k-1} = P_k.$$

7

Statement 3. $P_k P_m = P_m P_k = P_{\max(m,k)}$, and $Q_k Q_m = Q_m Q_k = Q_{\max(m,k)}$.

Start from

$$Q_k Q_{k-1} = Q_{k-1}^2 - \omega_k^{-1} Q_{k-1} y_k x_k^T A Q_{k-1}^2 = Q_{k-1} - \omega_k^{-1} Q_{k-1} y_k x_k^T A Q_{k-1} = Q_k$$

and complete the proof by induction.

Corollary 2. For $m \leqslant k$ it holds $P_k x_m = P_k P_m x_m = 0$ and $Q_k y_m = Q_k Q_m y_m = 0$.

Statement 4. For *biconjugate vectors* $u_k \overset{\text{def}}{=} P_{k-1} x_k$ and $v_k \overset{\text{def}}{=} Q_{k-1} y_k$ it holds

$$u_k^T A v_k = x_k^T A_{k-1} y_k \overset{\text{def}}{=} \omega_k,$$

since $A_{k-1} = A Q_{k-1} = A Q_{k-1}^2 = A_{k-1} Q_{k-1} = P_{k-1}^T A Q_{k-1}$. Also for $m \neq k$ it holds

$$u_m^T A v_k = x_m^T P_{m-1}^T A Q_{k-1} y_k = x_m^T A_{\max(m-1,k-1)} y_k = 0.$$

With $U_k \overset{\text{def}}{=} [u_1 \ldots u_k]$, $V_k \overset{\text{def}}{=} [v_1 \ldots v_k]$ and $\Omega_k \overset{\text{def}}{=} \text{diag}(\omega_1 \ldots \omega_k)$ we conclude

$$U_k^T A V_k = \Omega_k. \tag{6}$$

Corollary 3. For the valid Wedderburn process U_k and V_k have full rank.

Corollary 4. For $m \leqslant k$ it holds $P_k u_m = P_k P_{m-1} x_m = P_k x_m = 0$ and $Q_k v_m = 0$.

Statement 5. Each rank elimination step (4) writes without multiplication by A_{k-1} as follows

$$A_k = A_{k-1} - \frac{A v_k u_k^T A}{u_k^T A v_k} = A - \sum_{p=1}^{k} A v_p \omega_p^{-1} u_p^T A.$$

Now the rank-k *Wedderburn approximation* reads

$$\tilde{A}_k \overset{\text{def}}{=} A V_k \Omega_k^{-1} U_k^T A, \qquad A = A_k + \tilde{A}_k. \tag{7}$$

For the rank-r matrix A the residual $A_r = 0$ and the approximation $\tilde{A}_r$ is exact.

We see that each Wedderburn update (4) adds vector y_k to the kernel and x_k to the cokernel of the residual $A_k = A - \tilde{A}_k$. The approximation $\tilde{A}_k$, as a linear operator, interpolates A on subspaces spanned by X_k and Y_k, exactly

$$X_k^T \tilde{A}_k = X_k^T A, \quad \tilde{A}_k Y_k = A Y_k, \qquad U_k^T \tilde{A}_k = U_k^T A, \quad \tilde{A}_k V_k = A V_k. \tag{8}$$

This can be associated with the *Gaussian elimination*, that gives an approximation exact on certain rows and columns of the matrix. In this respect we refer to the process (4) as to the *Wedderburn elimination.*

In [5] it is shown how the proper choice of x_k, y_k reduces the Wedderburn elimination to the well-known matrix decompositions such as LU, QR, cross factorizations and Lanczos bidiagonalization. We consider another principle for the selection of vectors x_k, y_k at each step in the matrix case, that can be associated with Gaussian elimination with column or row pivoting. It produces a new method for matrix approximation, that is simply generalized to tensors while maintaining the convergence.

3.2 Pivoting in Wedderburn elimination

The idea behind the proposed choice of x_k, y_k is minimization of Frobenius norm of the residual, that is important when we deal with the full-rank matrix of a low ε-rank, i.e. that can be approximated by the low-rank matrix with the accuracy ε. As shown in [5], the minimization of residual w.r.t. unit x_k, y_k gives k-th singular vectors of A, that are not known in advance. We propose another minimization strategy, based on the following theorem.

Theorem 1. Consider Wedderburn step (3). Then

$$\text{for fixed } y, \qquad x_{opt} \overset{\text{def}}{=} \arg \min_{\|x\|=1} \left\| A - \frac{Ayx^TA}{x^TAy} \right\|_F = \frac{Ay}{\|Ay\|}; \tag{9}$$

$$\text{for fixed } x, \qquad y_{opt} \overset{\text{def}}{=} \arg \min_{\|y\|=1} \left\| A - \frac{Ayx^TA}{x^TAy} \right\|_F = \frac{A^Tx}{\|A^Tx\|}. \tag{10}$$

Proof. Since the valid Wedderburn step does not depend on scaling of x, y, for the fixed y we can constrain x to satisfy $x^TAy = 1$. Then

$$x_{opt} = \min_{x:\, x^TAy=1} \left\| (Ay)(A^Tx) - A \right\|_F^2 = \min_{x:\, x^TAy=1} \left\| (A^Tx)(Ay)^T - A^T \right\|_F^2.$$

The least squares problem solves by $(A^Tx)_{opt} = A^TAy/\|Ay\|^2$ and $x_{opt} = Ay/\|Ay\|^2$. After normalization we have (9), and (10) follows by substituting $A = A^T$. $\square$

Eqs. (9) and (10) show how to reach the fast decay of the residual in the Wedderburn process (4) either by choosing optimal x_k for given y_k or by choosing optimal y_k for given x_k. This can be associated with the column and row pivoting in the Gaussian elimination. Thus, we refer to the Wedderburn process (4) with arbitrary y_k and optimal x_k as to *Wedderburn elimination with column pivoting* (WCP) and to Wedderburn process with arbitrary x_k and optimal y_k as to *Wedderburn elimination with row pivoting* (WRP). The steps of WCP and WRP shortly read

$$\text{choose } y_k, \text{ set} \quad x_k = \frac{A_{k-1}y_k}{\|A_{k-1}y_k\|}, \quad A_k = \left(I - x_k x_k^T\right) A_{k-1}; \tag{WCP}$$

$$\text{choose } x_k, \text{ set} \quad y_k = \frac{A_{k-1}^T x_k}{\|A_{k-1}^T x_k\|}, \quad A_k = A_{k-1}\left(I - y_k y_k^T\right). \tag{WRP}$$

Theorem 2. For the Wedderburn elimination with column pivoting it holds

1. $X_k = [x_1 \ldots x_k]$ has orthonormal columns $X_k^T X_k = I$;

2. $P_k = P_k^T = I - X_k X_k^T$ is the projector on the subspace orthogonal to $\operatorname{span} X_k$;

3. biconjugate vectors $u_k \overset{\text{def}}{=} P_{k-1} x_k = x_k$.

For the Wedderburn elimination with row pivoting it holds

Algorithm 4: Wedderburn elimination with column pivoting (WCP)

Input: Matvec subroutine for the matrix A, tolerance parameter `tol`, accuracy ε

Output: Approximation $\tilde{A}$ with the accuracy $\|A - \tilde{A}\|_F \lesssim \varepsilon\|\tilde{A}\|_F$

Initialization: $k = 0$, $X_0 = [\varnothing]$, $B_0 = [\varnothing]$, $\texttt{nrm} = 0$

  1: **repeat**

  2:     $k := k + 1$, choose unit vector y_k

  3:     $x := Ay_k$; $x' := (I - X_{k-1}X_{k-1}^\mathsf{T})x$

  4:     **if** $\|x'\| < \texttt{tol}\|x\|$ **then** {Breakdown}

  5:        **return** $\tilde{A} = X_{k-1}B_{k-1}^\mathsf{T}$ {or repeat current iteration with another y_k}

  6:     **else**

  7:        $x_k := x'/\|x'\|$

  8:     **end if**

  9:     $b_k := A^\mathsf{T}x_k$, $\texttt{err} := \|b_k\|$, $\texttt{nrm}^2 := \texttt{nrm}^2 + \|b_k\|^2$

10:     $X_k := [X_{k-1}\ x_k]$, $B_k := [B_{k-1}\ b_k]$

11: **until** $\texttt{err} \leqslant \varepsilon\,\texttt{nrm}$

12: **return** $\tilde{A} = X_kB_k^\mathsf{T}$
---

1. $Y_k = [y_1 \ldots y_k]$ has orthonormal columns $Y_k^\mathsf{T}Y_k = I$;

2. $Q_k = Q_k^\mathsf{T} = I - Y_kY_k^\mathsf{T}$;

3. $v_k \stackrel{\text{def}}{=} Q_{k-1}y_k = y_k$.

Proof. Let us prove statements for WRP by induction. It is easy to check them for $k = 1$. Suppose they hold at step $k - 1$. Optimal choice of y_k by (10) reads

$$y = A_{k-1}^\mathsf{T}x_k = Q_{k-1}^\mathsf{T}A^\mathsf{T}x_k = (I - Y_{k-1}Y_{k-1}^\mathsf{T})A^\mathsf{T}x_k, \qquad y_k = y/\|y\|.$$

This shows $\|y_k\| = 1$, $Y_{k-1}^\mathsf{T}y_k = 0$ and the first statement follows for $Y_k := [Y_{k-1}\ y_k]$. By substituting $y_k = A_{k-1}^\mathsf{T}x_k/\|A_{k-1}^\mathsf{T}x_k\|$ in (5) we prove the second statement

$$Q_k = Q_{k-1} - \frac{Q_{k-1}y_kx_k^\mathsf{T}A_{k-1}}{x_k^\mathsf{T}A_{k-1}y_k} = Q_{k-1}\left(I - \frac{y_ky_k^\mathsf{T}}{y_k^\mathsf{T}y_k}\right) =$$
$$= \left(I - Y_{k-1}Y_{k-1}^\mathsf{T}\right)\left(I - y_ky_k^\mathsf{T}\right) = I - Y_{k-1}Y_{k-1}^\mathsf{T} - y_ky_k^\mathsf{T} = I - Y_kY_k^\mathsf{T}.$$

Finally, the last statement reads

$$v_k \stackrel{\text{def}}{=} Q_{k-1}y_k = Q_{k-1}\frac{Q_{k-1}^\mathsf{T}A^\mathsf{T}x_k}{\|A_{k-1}^\mathsf{T}x_k\|} = \frac{Q_{k-1}^2A^\mathsf{T}x_k}{\|A_{k-1}^\mathsf{T}x_k\|} = \frac{Q_{k-1}^\mathsf{T}A^\mathsf{T}x_k}{\|A_{k-1}^\mathsf{T}x_k\|} = y_k.$$

Statements for the WCP are proven in the same way. $\qquad\square$

    Based on this theorem, we propose a Wedderburn elimination algorithm with the column pivoting (WCP, see Alg. 4). The approximation is sought in the form $\tilde{A}_k = X_kB_k^\mathsf{T}$ with $B_k = A^\mathsf{T}X_k$, that follows from (8) and orthogonality of biconjugate vectors $U_k = X_k$, result of the Theorem 2. To explain the stopping criteria of Alg. 4, write

$$\texttt{nrm} \stackrel{\text{def}}{=} \|\tilde{A}_k\|_F = \|X_kX_k^\mathsf{T}A\|_F = \|A^\mathsf{T}X_k\|_F = \|B_k\|_F, \quad \texttt{err} \stackrel{\text{def}}{=} \|\tilde{A}_k - \tilde{A}_{k-1}\|_F = \|b_k\|_F,$$

where the first part estimates the norm of matrix by the norm of approximation and the second estimates the error of approximation by the norm of the update. The *breakdown* can occur if new vector x has the neglectable component x' orthogonal to the accumulated subspace X_{k-1}. This is regulated by the tolerance parameter `tol`, that could be chosen close to machine precision. We can try to fix the breakdown by another selection of y_k, or choose to terminate the algorithm.

The WRP version of algorithm follows by substituting $A = A^\mathsf{T}$.

3.3 Relation to SVD and Lanczos bidiagonalization

On each step of the WCP Alg. 4 we perform the Wedderburn elimination, choosing optimal 'pivot' x_k for given y_k. It is also important to select proper 'leading vectors' y_k to obtain faster convergence of approximation and avoid breakdowns. We could think about the maximization of $\omega_k = x_k^\mathsf{T} A_{k-1} y_k = \|A_{k-1} y_k\| = \|x'\|$. Solving exactly

$$y_k = \arg \max_{\|y\|=1} \|A_{k-1}y\| = \arg \max_{\|y\|=1} \left\| (I - X_{k-1}X_{k-1}^\mathsf{T})Ay \right\|, \tag{11}$$

we reduce the Wedderburn process to a sequence of best rank-one approximations with ω_k and x_k, y_k being singular values of A (sorted descending) and corresponding left and right singular vectors. Therefore, the WCP with selection of the leading vector by (11) is equivalent to the SVD and provides the best rank-r approximation in the Frobenius norm. This approach can be associated with the *full pivoting* in the Gaussian elimination. Each maximization problem (11) can be accurately solved by power iterations using only matvec operations, but in matrix case this approach generally is considered as quite expensive and faster alternatives are used.

One of them in the Lanczos bidiagonalization, see [12] and Alg. 5. It generates bases $X_k = [x_1 \dots x_k]$ and $Y_k = [y_1 \dots y_k]$ such that $X_k^\mathsf{T} X_k = I$, $Y_k^\mathsf{T} Y_k = I$ and the matrix $X_k^\mathsf{T} A Y_k$ is bidiagonal.

The following theorem shows that the Lanczos bidiagonalization is similar to the Alg. 4 if the leading vector is selected as follows

$$y_{k+1} = \frac{A_{k-1}^\mathsf{T} x_k}{\|A_{k-1}^\mathsf{T} x_k\|} = \frac{A^\mathsf{T} x_k}{\|A^\mathsf{T} x_k\|}. \tag{12}$$

This value is well defined, since

$$A_{k-1}^\mathsf{T} x_k = A^\mathsf{T} P_{k-1} x_k = A^\mathsf{T} (I - X_{k-1}X_{k-1}^\mathsf{T}) x_k = A^\mathsf{T} x_k = b_k,$$

and $\|A_{k-1}^\mathsf{T} x_k\| \overset{\text{def}}{=}$ err vanishes only when the stopping criteria err $< \varepsilon$ nrm is met and the next leading vector y_{k+1} is not required.

Theorem 3. Vectors $X^{\{lnc\}} = [x_1^{\{lnc\}}, \dots, x_k^{\{lnc\}}]$ of the Lanczos process (Alg. 5) initialized by x_0, coincide with vectors $X_k^{\{wcp\}} = [x_1^{\{wcp\}}, \dots, x_k^{\{wcp\}}]$ generated by WCP Alg. 4 that starts from $y_1 = A^\mathsf{T} x_0$ and chooses y_k as proposed by (12), providing both algorithms do not meet breakdowns.

Algorithm 5: [12] Lanczos bidiagonalization

Initialization: $y_0 = 0, \beta_0 = 0$, unit vector x_0

    for $k = 1, 2, \ldots$ **do**

        $y := A^\mathsf{T} x_{k-1}; \quad y' := y - \beta_{k-1} y_{k-1}, \quad \alpha_k := \|y'\|, \quad y_k := y'/\|y'\|$

        $x := A y_k; \quad x' := x - \alpha_k x_k, \quad \beta_k := \|x'\|, \quad x_k := x'/\|x'\|$

    end for

Proof. If no breakdowns are met, then

$$\operatorname{span} X_k^{\{\mathtt{wcp}\}} = \operatorname{span}\{A y_1, (AA^\mathsf{T}) A y_1, \ldots, (AA^\mathsf{T})^{k-1} A y_1\},$$
$$\operatorname{span} X_k^{\{\mathtt{lnc}\}} = \operatorname{span}\{(AA^\mathsf{T}) x_0, (AA^\mathsf{T})^2 x_0, \ldots, (AA^\mathsf{T})^k x_0\}.$$

If $y_1 = A^\mathsf{T} x_0$ then $\operatorname{span} X_k^{\{\mathtt{wcp}\}} = \operatorname{span} X_k^{\{\mathtt{lnc}\}}$ for every k. Since columns of $X_k^{\{\mathtt{wcp}\}}$ are orthonormal (Theorem 2), as well as $X_k^{\{\mathtt{lnc}\}}$, we conclude $X_k^{\{\mathtt{wcp}\}} = X_k^{\{\mathtt{lnc}\}}$. $\qquad\square$

Thus, in terms of x_k, the WCP gives the same result as the Lanczos bidiagonalization. However, the sequence y_k in the WCP differs from the one of the Lanczos bidiagonalization. First, we note that

$$y = A_{k-1}^\mathsf{T} A_{k-1} y_k = A^\mathsf{T} P_{k-1} P_{k-1}^\mathsf{T} A y_k = A^\mathsf{T} P_{k-1} A y_k = Q_{k-1}^\mathsf{T} A^\mathsf{T} A y_k, \quad y_{k+1} = \frac{y}{\|y\|}$$

and $\operatorname{span} Y_k = \operatorname{span}\{y_1, (A^\mathsf{T} A) y_1, \ldots, (A^\mathsf{T} A)^{k-1} y_1\}$. Sequence y_k is 'almost orthogonal', i.e. for $m \leqslant k - 2$ it holds

$$y_k^\mathsf{T} y_m = \frac{y_{k-1}^\mathsf{T} A^\mathsf{T} A Q_{k-2} y_m}{\|A^\mathsf{T} A y_{k-1}\|} = 0$$

since $Q_{k-2} y_m = 0$ by the Corollary 2. Also, for X_k and Y_k generated by the WCP, the matrix $X_k^\mathsf{T} A Y_k$ is tridiagonal, i.e.

$$x_m^\mathsf{T} A y_k = y_{m+1}^\mathsf{T} y_k = 0, \qquad \text{for} \quad m \notin \{k - 2, k - 1, k\}.$$

Remark 1. For $k \geqslant 3$ the vector $X_{k-1}^\mathsf{T} x = X_{k-1}^\mathsf{T} A y_k$ on Step 3 of Alg. 4 has only two last nonzero components. Hence, the orthogonalization step can be simplified to

$$x := A y_k, \quad x' := (I - x_{k-2} x_{k-2}^\mathsf{T} - x_{k-1} x_{k-1}^\mathsf{T}) x.$$

This allows the short recursion for the orthogonalization of x_k in the WCP Alg. 4, as it is done in the Lanczos bidiagonalization Alg. 5. We summarize this version of the WCP in the Alg. 6. In the machine arithmetic the orthogonality can be violated by roundoff errors and the reorthogonalization is required. The convergence of the Lanczos bidiagonalization method is well studied and although this method can have breakdowns, it is considered to converge from 'almost every' initial vector [29, 13, 16].

Algorithm 6: WCP with Lanczos-like selection of leading vector

---

Input: Matvec subroutine for matrix A, tolerance parameter `tol`, accuracy ε
Output: Approximation $\tilde{A}$ with accuracy $\|A - \tilde{A}\|_F \lesssim \varepsilon \|\tilde{A}\|_F$
Initialization: $k = 0$, $\quad x_0 = x_{-1} = 0$, $\quad \tilde{A}_0 = 0$, $\quad \mathtt{nrm} = 0$, unit vector y_1

 1: **repeat**
 2:     $k := k + 1$, $\quad x := A y_k$; $\quad x' := (I - x_{k-2} x_{k-2}^T - x_{k-1} x_{k-1}^T) x$
 3:     **if** $\|x'\| < \mathtt{tol}\|x\|$ **then** {Breakdown}
 4:         **return** $\tilde{A} = \tilde{A}_{k-1}$
 5:     **else**
 6:         $x_k := x'/\|x'\|$
 7:     **end if**
 8:     $y_{k+1} := A^T x_k$, $\quad \mathtt{err} := \|y_{k+1}\|$, $\quad \mathtt{nrm}^2 := \mathtt{nrm}^2 + \|y_{k+1}\|^2$
 9:     $\tilde{A}_k = \tilde{A}_{k-1} + x_k y_{k+1}^T$
10: **until** $\mathtt{err} \leqslant \varepsilon\,\mathtt{nrm}$
11: **return** $\tilde{A} = \tilde{A}_k$

---

4 Tensor approximation using Wedderburn rank reduction

4.1 Computing dominant mode subspaces by WCP algorithm

We are ready to propose the extension of the Wedderburn elimination to the tensor case. For the Tucker approximation we are to approximate the dominant subspaces by U, V, W that contain much of information about mode vectors of tensor. In [6] this is done by SVD applied to the unfoldings of an $n_1 \times n_2 \times n_3$ tensor $\mathbf{A} = [a_{ijk}]$. They are matrices

$$A^{(1)} = [a_{i,jk}^{(1)}], \quad A^{(2)} = [a_{j,ki}^{(2)}], \quad A^{(3)} = [a_{k,ij}^{(3)}], \qquad a_{i,jk}^{(1)} = a_{j,ki}^{(2)} = a_{k,ij}^{(3)} = a_{ijk}, \qquad (13)$$

of size $n_1 \times n_2 n_3$, $n_2 \times n_1 n_3$ and $n_3 \times n_1 n_2$, that consist of columns, rows and tube fibres of $\mathbf{A}$, respectively. Left singular vectors after appropriate truncation give Tucker factors U, V, W, and the core is found by (1). Since the SVD is applied to tensors that are given as full array of elements, the computation costs $\mathcal{O}(n^4)$ and can not is not possible for large tensors (even sparse or structured). We approximate dominant subspaces of mode vectors applying the Wedderburn elimination to the unfoldings.

We aim for algorithms that compute the rank-(r_1, r_2, r_3) approximation of an $n_1 \times n_2 \times n_3$ tensor using $\mathcal{O}(r^2)$ tenvecs and $\mathcal{O}(n)$ additional operations. This is asymptotically equal to the cost of the MKR Alg. 1, if the core tensor is computed. With this restriction for $n_1 \times n_2 n_3$ unfolding $A = A^{(1)} = [a_{i;jk}]$ we can not compute $x = A\mathbf{y}$ and $\mathbf{y} = A^T x$ for arbitrary vectors $\mathbf{y}$ of size $n_2 n_3$ and x of size n_1, since these operations require $\mathcal{O}(n^2)$ storage and n tenvecs. To develop the Krylov-type methods and stay within the linear complexity in mode size, we should use only those operations, that can be accomplished

---

Algorithm 7: Wedderburn elimination for dominant subspace computation

Input: Tenvec subroutine for $\mathbf{A}$, tolerance `tol`, accuracy ε, maximum size r_{max}.

Output: Mode subspace U for Tucker approximation $\tilde{\mathbf{A}}$ such that $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F \lesssim \varepsilon\|\tilde{\mathbf{A}}\|_F$

Initialization: $X_0 = [\varnothing]$, $\quad$ nrm $= 0$, $\quad$ k $= 0$

1: **repeat**
2: $\quad$ k := k $+$ 1, choose unit vectors y_k, z_k (see Section 4.2)
3: $\quad$ $x := \mathbf{A}y_k z_k$; $\quad x' := (I - X_{k-1}X_{k-1}^T)x$
4: $\quad$ **if** $\|x'\| < $ `tol`$\|x\|$ **then** {Breakdown}
5: $\quad\quad$ **return** $U = X_{k-1}$ {or repeat current iteration with another y_k, z_k}
6: $\quad$ **else**
7: $\quad\quad$ $x_k := x'/\|x'\|$; $\quad X_k := [X_{k-1}\; x_k]$
8: $\quad$ **end if**
9: $\quad$ Choose unit z randomly
10: $\quad$ **for** k $= 1, \ldots, p_{pow}$ **do** {Power iterations to approximate $\mathbf{A} \times_1 x_k^T \approx: \sigma y z^T$}
11: $\quad\quad$ $y := (\mathbf{A} \times_1 x_k^T)\; z = \mathbf{A} \times_1 x_k^T \times_3 z^T = \mathbf{A}zx_k$, $\quad \sigma := \|y\|$, $\quad y := y/\|y\|$
12: $\quad\quad$ $z := (\mathbf{A} \times_1 x_k^T)^T y = \mathbf{A} \times_1 x_k^T \times_2 y^T = \mathbf{A}x_k y$, $\quad \sigma := \|z\|$, $\quad z := z/\|z\|$
13: $\quad$ **end for**
14: $\quad$ err $:= \sigma$, $\quad$ nrm$^2 :=$ nrm$^2 +$ err^2
15: **until** err $\leqslant \varepsilon$ nrm or k $= r_{max}$
16: **return** $U = X_k$

---

by small number of tenvecs. For instance, $x = A\mathbf{y}$ is substituted by

$$x = A(y \otimes z) = \mathbf{A} \times_2 y^T \times_3 z^T = \mathbf{A}yz,$$

that means that we will use only those 'long vectors' $\mathbf{y}$ that are the tensor product $y \otimes z$ of some y of size n_2 and z of size n_3. Evaluation of $\mathbf{y} = A^T x = \mathbf{A} \times_1 x$ is also infeasible and we substitute it with the approximation

$$y \otimes z \approx \mathbf{A} \times_1 x,$$

that means that we develop the algorithms that give accurate tensor approximation, using only certain rank-one approximation of $\mathbf{A} \times_1 x$ instead of the precise result.

In the matrix case the difference between column and row pivoting in the Wedderburn elimination is not significant, since the properties of column basis in the WCP coincide with ones of row basis in the WRP and vice versa. In the tensor case with imposed restriction the multiplication $A(y \otimes z)$ is accurate, but the multiplication $A^T x = \mathbf{A} \times_1 x^T$ is always done approximately with an error of truncation to rank one. Considering this difference, we use the WCP algorithm for the tensor case, since good properties of X_k given by Theorem 2 persist here. Using the direct analogy with the WCP Alg. 4 for matrices, we propose a method to derive dominant mode-1 subspace U, see Alg. 7.

Each iteration begins with the choice of the 'long' leading vector in the form $y_k \otimes z_k$. It can be done arbitrarily, and we propose some good strategies in the section 4.2. The new direction x and basis vector x_k appear exactly like in the matrix case. To terminate the method, we can use one or both of the stopping criteria:

- fix maximum number of iterations, i.e. desired size of basis U by r_{max},

- find basis U that allows approximation of $\mathbf{A}$ with relative accuracy ε.

In the WCP Alg. 4 the error was estimated by the norm of vector $b_k = A^\top x_k$. In the tensor algorithm we should estimate the Frobenius norm of the matrix $\mathbf{A} \times_1 x_k$ instead, but it can not be evaluated by a small number of tenvecs and we substitute it by the spectral norm $err \overset{\text{def}}{=} \|\mathbf{A} \times_1 x_k\|_2$. To estimate this, we use the power method for $n_2 \times n_3$ matrix $\mathbf{A} \times_1 x_k^\top$. It is initialized by some randomly chosen unit vector z of size n_3. In the power method, σ converges to the maximum singular value of matrix $B_k = \mathbf{A} \times_1 x_k^\top$ and $\tilde{B}_k = \sigma y z^\top$ converges to the best rank-one approximation of B_k. Since the high precision is not necessary for the error estimation, we can satisfy with the fixed small number p_{pow} of power iteration steps. If only the fixed-rank stopping criteria is desired, power iterations on Steps 9-13 of Alg. 7 can be omitted.

As well as in the matrix case, we can meet with the *breakdown* if new vector x have almost zero (neglectable in the machine precision) component x' orthogonal to the subspace X_{k-1}. We can try to fix it by repeating the current step with another selection of y_k, z_k, or choose to terminate the algorithm.

Mode-2 and mode-3 bases V and W can be computed by the same algorithm after obvious permutation of modes. Directly from the Statement 5 in matrix case we derive the following theorem, that proves the convergence of tensor methods based on Alg. 7 for the exact-rank case.

Theorem 4. For the tensor $\mathbf{A}$ with mode sizes n_1, n_2, n_3 and ranks r_1, r_2, r_3, three applications of Alg. 7 return bases U, V and W of sizes $n_1 \times r_1$, $n_2 \times r_2$ and $n_3 \times r_3$ that allow the Tucker decomposition $\mathbf{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ with core $\mathbf{G}$ given by (1), providing that computations are not terminated by breakdowns.

Remark 2. The convergence in the exact-rank case is guaranteed for any choice of the leading vectors y_k and z_k that does not lead to the breakdown.

In the section 4.2 we propose a number of strategies for the selection of the leading vectors, that are based on the different optimization ideas and lead to the methods with different complexity and convergence properties. However, the convergence in the exact-rank case, which we consider as the necessary requirement, persists for all methods based on the Alg. 7.

Remark 3. The minimal Krylov recursion Alg. 1 can be also considered as variant of Alg. 7 with the special choice of leading vectors.

To achieve the reduction of mode-1 rank after the first iteration of the Alg. 7 we should approximate the tensor by

$$\mathbf{A} \approx \tilde{\mathbf{A}}_1 = x_1 \otimes B_1 \qquad \text{with} \qquad B_1 = \mathbf{A} \times_1 x_1. \tag{14}$$

This does not comply the restrictions imposed to reach the linear complexity. Therefore, we consider Alg. 7 as a method to generate an approximation of the dominant mode

subspaces that is guaranteed to converge in r steps for the tensor with the mode rank r, but that does not reduce the mode ranks by one on each iteration.

Remark 4. The mode ranks of the rank-(r_1, r_2, r_3) tensor in general can not be reduced by the elimination of the rank-$(1, 1, 1)$ approximation.

To take use of the approximation (14), we can further approximate B_1 by low-rank (or even rank-one) format. To do this by tenvec operations, we can use 'internal' Wedderburn elimination steps for B_1. This approach directly follows the basic idea of [25], where the cross approximation method is generalized to 3-tensors. Vectors x_k can be associated with the mode fibres selected in the tensor, and B_k with the slices, for which the internal cross approximation scheme is used. As well as in the matrix case, we emphasize the very important link between the Krylov subspaces approach and cross approximation methods, that is established by the Wedderburn framework.

In the following we restrict the discussion to only rank-one approximation of B_k and pay by losing the mode rank reduction property, but result in the methods that are more efficient since they have 'symmetric' behaviour in respect to different modes.

4.2 Selecting leading vector

By the Theorem 4, in Alg. 7 the every choice of the leading vectors y_k, z_k that does not lead to the breakdown, ensures the convergence in the exact-rank case. However the different choices of leading vectors result in the approximations with different accuracy until the exact representation is found. Therefore, we should choose the leading vectors in a clever way, that ensures the fast convergence to the dominant subspaces and is computationally feasible. In the following we propose four strategies that lead to different maximization problems and result in different complexity estimates and convergence properties.

4.2.1 SVD-like strategy (Wsvd)

We can apply Alg. 7 three times and compute U, V, W in the completely independent processes, even using three processors on a distributed memory system. In algorithm for $U = X_k$ the best way to keep from breakdowns is to choose $y_k \otimes z_k$ that maximizes the orthogonal component $\|x'\|$.

$$
\begin{aligned}
y_k, z_k &= \arg \max_{\|y\|=\|z\|=1} \left\| (I - X_{k-1} X_{k-1}^{\mathsf{T}})(\mathbf{A} y z) \right\| = \arg \max_{\|y\|=\|z\|=1} \|\mathbf{B} y z\|, \\
\mathbf{B} &= \mathbf{A} \times_1 (I - X_{k-1} X_{k-1}^{\mathsf{T}}).
\end{aligned}
\tag{15}
$$

This is the direct analogy with the SVD approach (11) in the matrix case. The difference is that resulted x_k and $y_k \otimes z_k$ are not singular vectors of the unfolding $A = A^{(1)}$, since the maximization is done w.r.t. 'long' right vectors with tensor product structure. This is the reason why the best tensor rank-(r, r, r) approximation can not be computed by the simultaneous elimination of the best rank-$(1, 1, 1)$ approximations.

Nevertheless with this choice we are safe from breakdowns. Zero-valued orthogonal component $\|x'\|$ appears only if $\max_{\|y\|=\|z\|=1}\|\mathbf{B}yz\| = 0$ and hence $\mathbf{B} = 0$, which means that the *exact* representation $\tilde{\mathbf{A}}_k \stackrel{\text{def}}{=} \mathbf{A} \times_1 (X_k X_k^\mathsf{T})$ with mode-1 rank $r_1 = k$ is computed for $\mathbf{A}$. The following theorem shows that 'machine precision breakdown' also happens only when the approximation is of the machine precision accuracy.

Theorem 5. If the breakdown $\|x'\| < \mathtt{tol}\|x\|$ is met on the step $k+1$ of the Alg. 7, than the computed subspace X_k provides the approximation $\tilde{\mathbf{A}}_k = \mathbf{A} \times_1 (X_k X_k^\mathsf{T})$ with the Tucker rank $r_1 = k$ and accuracy $\|\mathbf{A} - \tilde{\mathbf{A}}_k\|_2 < \mathtt{tol}\|\mathbf{A}\|_2$ in spectral norm.

Proof. On the step $k+1$ we choose $y_{k+1}, z_{k+1} = \arg\max_{\|y\|=\|z\|=1}\|\mathbf{B}yz\|$. The norm of the orthogonal component reads

$$\|x'\| = \|\mathbf{B}y_{k+1}z_{k+1}\| = \max_{\|y\|=\|z\|=1}\|\mathbf{B}yz\|$$

$$= \max_{\|y\|=\|z\|=1}\max_{\|x\|=1}(\mathbf{B}yz, x) = \max_{\|x\|=\|y\|=\|z\|=1}\langle \mathbf{B}, x \otimes y \otimes z \rangle \stackrel{\text{def}}{=} \|\mathbf{B}\|_2.$$

Also for $x = \mathbf{A}y_{k+1}z_{k+1}$ it holds

$$\|x\| = \|\mathbf{A}y_{k+1}z_{k+1}\| = \max_{\|x\|=1}(\mathbf{A}y_{k+1}z_{k+1}, x) \leqslant \max_{\|x\|=\|y\|=\|z\|=1}\langle \mathbf{A}, x \otimes y \otimes z \rangle \stackrel{\text{def}}{=} \|\mathbf{A}\|_2.$$

Now the breakdown criteria $\|x'\| < \mathtt{tol}\|x\|$ gives $\|\mathbf{B}\|_2 < \mathtt{tol}\|\mathbf{A}\|_2$. Finally we write

$$\mathbf{B} = \mathbf{A} \times_1 (I - X_k X_k^\mathsf{T}) = \mathbf{A} - \mathbf{A} \times_1 (X_k X_k^\mathsf{T}) = \mathbf{A} - \tilde{\mathbf{A}}_k,$$

that completes the proof since mode-1 rank of $\tilde{A}_k$ is equal to $r_1 = \mathrm{rank}\,X_k = k$. $\qquad\square$

Corollary 5. Alg. 7 with the SVD-like strategy (15) applied to tensor $\mathbf{A}$ with mode ranks r_1, r_2, r_3 computes bases U, V, W that allow the exact representation $\mathbf{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ after r_1, r_2 and r_3 iterations, respectively.

Note that since $X_k = [X_{k-1}\ \tilde{x}_k]$ is orthogonal, it holds

$$\mathbf{A} \times_1 x_k^\mathsf{T} = \mathbf{A} \times_1 \big((I - X_{k-1}X_{k-1}^\mathsf{T})x_k\big)^\mathsf{T} = \big(\mathbf{A} \times_1 (I - X_{k-1}X_{k-1}^\mathsf{T})\big) \times_1 x_k^\mathsf{T} = \mathbf{B} \times_1 x_k^\mathsf{T}.$$

The maximization problem (15) can be solved by the ALS Alg. 2 applied for $\mathbf{B}$. In this case power iterations for matrix $\mathbf{A} \times_1 x_k^\mathsf{T}$ on Step 9-13 of Alg. 7 can be omitted, since the error estimate $\mathtt{err} = \|\mathbf{A} \times_1 x_k^\mathsf{T}\|_2 = \|\mathbf{B} \times_1 x_k^\mathsf{T}\|_2$ is actually computed in the ALS iterations.

On the step k of the Wedderburn method inner ALS iterations cost $3p_{\mathrm{als}}$ tenvecs and $\mathcal{O}(p_{\mathrm{als}}nk)$ operations for the orthogonalization. This summarizes to $3p_{\mathrm{als}}r$ tenvecs and $\mathcal{O}(p_{\mathrm{als}}nr^2)$ additional operations for one dominant subspace.

4.2.2 Lanczos-like strategy (Wlnc)

We can also use the analogy with the Lanczos choice (12) by taking unit $y_k \otimes z_k \approx \mathbf{A} \times_1 x_k^\mathsf{T} / \|\mathbf{A} \times_1 x_k^\mathsf{T}\|$. This leads to the dual maximization problem

$$
\begin{aligned}
y_k, z_k &= \arg \max_{\|y\|=\|z\|=1} \left\| \mathbf{A} \times_1 x_k^\mathsf{T} \times_2 y^\mathsf{T} \times_3 z^\mathsf{T} \right\| = \arg \max_{\|y\|=\|z\|=1} \left\| y^\mathsf{T} B z \right\| \\
B &= \mathbf{A} \times_1 x_k^\mathsf{T}.
\end{aligned}
\tag{16}
$$

The solution can be accomplished by p_{pow} steps of the power iteration method applied to matrix B. On the step k of the Wedderburn method it costs $2p_{\mathrm{pow}}$ tenvecs. Note that the maximization problem (16) is actually solved on Steps 9-13 of Alg. 7 by power iterations that computes best rank-1 approximation of $\mathbf{A} \times_1 x_k^\mathsf{T}$ to estimate the norm of the residual. Thus, the Wlnc pivoting strategy requires only to set $y_k := y$ and $z_k := z$ after power iterations as new vectors of the Wedderburn process.

4.2.3 Restricted SVD-like strategy (WsvdR)

Three Wedderburn elimination algorithms can be used to extend all mode bases *simultaneously*. Suppose $k - 1, l$ and m steps were done to compute mode subspaces X_{k-1}, Y_l, Z_m. Then on the step k of the Wedderburn process for the mode-1 subspace we can make use of $V = Y_l$ and $W = Z_m$ by *restricting* the maximization (15) to the tensor product of these subspaces. Therefore, we take $y_k = Y_l \hat{y}_k, z_k = Z_m \hat{z}_k$ and solve

$$
\begin{aligned}
\hat{y}_k, \hat{z}_k &= \arg \max_{\|\hat{y}\|=\|\hat{z}\|=1} \left\| (I - X_{k-1} X_{k-1}^\mathsf{T}) \left(\mathbf{A}(Y_l \hat{y})(Z_m \hat{z}) \right) \right\| = \arg \max_{\|\hat{y}\|=\|\hat{z}\|=1} \left\| \mathbf{B} \hat{y} \hat{z} \right\| \\
\mathbf{B} &= \mathbf{A} \times_1 (I - X_{k-1} X_{k-1}^\mathsf{T}) \times_2 Y_l^\mathsf{T} \times_3 Z_m^\mathsf{T}.
\end{aligned}
\tag{17}
$$

This maximization problems exactly matches the (2)!

Remark 5. The optimized MKR is a variant of the Wedderburn process for tensors with restricted SVD-like strategy of pivoting.

The WsvdR approach can lead to the slow convergence or breakdowns at the first iterations, but when X_k, Y_l, Z_m become larger, chances to meet the breakdown vanish, since (17) becomes close to the unrestricted maximization (15). Numerical experiments provided in the section 5.2 show that sometimes the restricted SVD strategy gives even better results than unrestricted SVD strategy. The reason is probably that with this restrictions the ALS iterations are not caught in the local minima.

Complexity on the step k for the mode-1 subspace is $3p_{\mathrm{als}}$ tenvecs and $\mathcal{O}(p_{\mathrm{als}} n k)$ operations for the orthogonalization, total complexity is $9p_{\mathrm{als}} r$ tenvecs and $\mathcal{O}(p_{\mathrm{als}} n r^2)$ additional operations.

4.2.4 Restricted Lanczos-like strategy (WlncR)

Algorithm 8: Wedderburn with restricted Lanczos-like pivoting strategy (WlncR)

Input: Tenvec subroutine for the tensor $\mathbf{A}$, tolerance `tol`, accuracy ε

Output: Approximation $\tilde{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W$ such that $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F \lesssim \|\tilde{\mathbf{A}}\|_F$

Initialization: Unit vectors u, v, w, $updX = updY = updZ = \mathbf{true}$.

 1: $x_1 = \mathbf{A}vw/\|\mathbf{A}vw\|$; $y_1 = \mathbf{A}wu/\|\mathbf{A}wu\|$; $z_1 = \mathbf{A}uv/\|\mathbf{A}uv\|$, $k = l = m = 1$

 2: $X_1 = [x_1], Y_1 = [y_1], Z_1 = [z_1]$, $\mathbf{G} = \mathbf{A} \times_1 x_1^T \times_2 y_1^T \times_3 z_1^T$; $nrm = \|\mathbf{G}\|_F$

 3: **while** $updX$ or $updY$ or $updZ$ **do**

 x: {Proceed with new vector x if required}

 4:    **if** $updU$ **then**

 5:       For $B = \mathbf{G}(k, :, :)$ solve $B \approx: \tilde{B} = b\hat{y}_k\hat{z}_k^T$ {best rank-one approximation}

 6:       $y_k := Y_l\hat{y}_k$, $z_k := Z_m\hat{z}_k$; $x := \mathbf{A}y_kz_k$; $x' := (I - X_kX_k^T)x$

 7:       **if** $\|x'\| < $ `tol`$\|x\|$ **then** {Breakdown}

 8:         $updX := \mathbf{false}$ {or repeat current iteration step with another y_k, z_k}

 9:       **else**

10:         $x_{k+1} := x'/\|x'\|$; $X_{k+1} := [X_k \; x_{k+1}]$

11:         Enlarge $\mathbf{G}$ by $\mathbf{G}(k+1, :, :) := \mathbf{A} \times_1 x_{k+1}^T \times_2 Y_l^T \times_3 Z_m^T$

12:         $err := \|\mathbf{G}(k+1, :, :)\|_F$, $nrm^2 := nrm^2 + err^2$, $k := k+1$

13:         **if** $err < \varepsilon\, nrm$ **then** {Convergence}

14:           $updX := \mathbf{false}$

15:         **end if**

16:       **end if**

17:    **end if**

 y: {Proceed with new vector y if required}

 z: {Proceed with new vector z if required}

18: **end while**

19: **return** $U = X_k, V = Y_l, W = Z_m$, $\tilde{A} = \mathbf{G} \times_1 U \times_2 V \times_3 W$

Finally, we combine the Lanczos-like selection of leading vectors (16) and the restricted maximization.

$$
\begin{aligned}
\hat{y}_k, \hat{z}_k &= \arg \max_{\|\hat{y}\| = \|\hat{z}\| = 1} \left\| (Y_l\hat{y})^T (\mathbf{A} \times_1 x_k^T)(Z_m\hat{z}) \right\| = \arg \max_{\|\hat{y}\| = \|\hat{z}\| = 1} \left\| \hat{y}^T B\hat{z} \right\| \\
B &= \mathbf{A} \times_1 x_k^T \times_2 Y_l^T \times_3 Z_m^T, \qquad y_k = Y_l\hat{y}_k, \quad z_k = Z_m\hat{z}_k.
\end{aligned}
\tag{18}
$$

If only bases U, V, W for the approximation are required, the maximization can be accomplished by p_{pow} steps of power iteration method applied to the matrix B. At every step of the Wedderburn method it requires $2p_{pow}$ tenvecs. But if the core tensor is also desired, the efficiency can be highly improved by precomputing B as $l \times m$ matrix. Comparing (18) and (1) we note that B is exactly the last mode-1 slice from the optimal core for the approximation of $\mathbf{A}$ in bases $U = X_k, V = Y_l, W = Z_m$

$$
\mathbf{G}(k, :, :) = \mathbf{A} \times_1 (U(:, k))^T \times_2 V^T \times_3 W^T = \mathbf{A} \times_1 x_k^T \times_2 Y_l^T \times_3 Z_m^T = B.
$$

Thus if we need the core $\mathbf{G}$, we prefer to compute it slice-by-slice in the Wedderburn process. Then we can apply standard matrix tools (SVD or cross methods) to find the

Table 1. Complexity of the algorithms for rank-(r_1, r_2, r_3) tensor approximation

name	description	output	tenvecs
MKR	Min. Krylov recursion [30] Alg. 1	U, V, W	$3r$
Wsvd	Alg. 7 with strategy (15)	U, V, W	$9p_{als}r + 3r$
Wlnc	Alg. 7 with strategy (16)	U, V, W	$6p_{pow}r + 3r$
WsvdR	Alg. 7 with strategy (17), Alg. 3	U, V, W	$9p_{als}r + 3r$
WlncR	Alg. 7 with strategy (18), Alg. 8	U, V, W	$6p_{pow}r + 3r$
WlncR	$-//-$	$U, V, W, \mathbf{G}$	$r^2 + 3r$

best rank-one approximation of B and solve (18) without additional tensor operations. This version of the Wedderburn elimination algorithm for tensors is summarized in the Alg. 8.

4.3 Comparison of the algorithms

To compare the versions of the proposed algorithm, we give their complexities in Table 1. If only subspaces U, V, W for Tucker rank-(r_1, r_2, r_3) approximation are required, than the minimal Krylov recursion [30] is fastest in terms of number of tenvecs used, since it requires only $3r$ tensor operations. All versions of the Wedderburn elimination Alg. 7 also require $\mathcal{O}(r)$ tenvecs, with factor depending only on the selected number of ALS or power iterations. Note that for $p_{als} = p_{pow}$, Lanczos-like pivoting strategy and SVD-like pivoting take roughly the same time, but SVD-like pivoting is guaranteed to be free from breakdowns. This differ tensor algorithms from the matrix case.

If the core $\mathbf{G}$ for Tucker approximation is required, we generally need additional r^2 tenvecs to evaluate it by (1). In this case, the fastest version of the Wedderburn elimination algorithm is the WlncR Alg. 8. It uses *exactly* the same number of tenvecs as the MKR, but shows better convergence in numerical experiments.

5 Numerical examples

The numerical experiments presented in this section were performed on the Intel Xeon Quad-Core E5504 CPU running at 2.00 GHz. In the section 5.1 we use MATLAB version 7.7.0, in sections 5.2, 5.3 we use Intel Fortran compiler version 11.1 and BLAS/LAPACK routines provided by the MKL library.

5.1 Sparse tensors

The tensor decomposition of a sparse large-dimensional arrays is an important tool in the network analysis, that is widely used now in the information science, sociology

Figure 1. Approximation accuracy for the Caltech Facebook graph
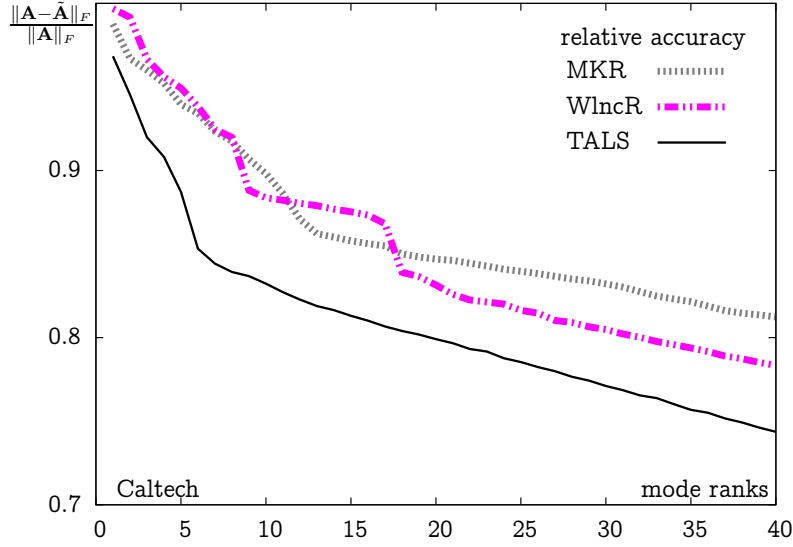
We approximate the $597 \times 597 \times 8^2$ tensor using three algorithms and plot relative accuracy in the Frobenius norm for different values of mode ranks. TALS is the Tucker-ALS, MKR is the minimal Krylov recursion, WlncR is the Wedderburn elimination with the Lanczos-like restricted pivoting.

and many other disciplines. We consider one example from [34], where a very nice introduction to network analysis is given.

The example is the graph of the Facebook social network from Caltech University, coupled with the dormitory information. The graph represents relations between $n = 597$ people living in $m = 8$ dorms by sparse $n \times n \times m \times m$ 4-tensor $\mathbf{A}$. Every entry $a_{ijpq} = 1$ means that the person number i lives in the dorm number p and links to the person number j, who lives in the dorm q. There are 25646 unit elements in $\mathbf{A} = [a_{ijpq}]$, and all others are zeroes. Since the discussed methods apply to 3-tensors, we join dorm indices p and q in multi-index and consider $\mathbf{A} = [a_{i,j,pq}]$ as the 3-tensor of size $n \times n \times m^2$. The relative accuracy of the approximation in the Frobenius norm is shown on Fig. 1. We compare three methods: the Tucker-ALS [22, 7] (MATLAB implementation from Tensor Toolbox [1]), the minimal Krylov recursion (Alg. 1) and the Wedderburn elimination method with Lanczos-like restricted pivoting (Alg. 8). We note that the WlncR Alg. 1 converges slowly on the first iterations, when dimensions of accumulated bases are small, imposing serious restrictions in the maximization (18). However, for the larger values of ranks, the WlncR becomes more accurate than the MKR method.

The Tucker-ALS is the most accurate but computationally demanding method. On each Tucker-ALS iteration the Tucker factors are subsequently updated as follows. With two factors fixed, for example V and W, we compute the $n_1 \times r_2 \times r_3$ tensor

$$\mathbf{B} = \mathbf{A} \times_2 V^\mathsf{T} \times_3 W^\mathsf{T}, \qquad B = B^{(1)} \tag{19}$$

21

Then for the $n_1 \times r_2 r_3$ unfolding $B = B^{(1)}$ we find the approximation $B \approx \tilde{B} =: UG$ with the $n_1 \times r_1$ orthogonal matrix U that is a new Tucker factor and the $r_1 \times r_2 r_3$ matrix G that is then reshaped to new $r_1 \times r_2 \times r_3$ core tensor $\mathbf{G}$. The rank-r_1 approximation of B can be computed by the SVD or cross approximation methods (c.f. [36]). The evaluation of (19) requires $r_2 r_3$ tenvecs, which results in $3r^2$ tenvecs on each iteration, much more than the complexity of the MKR and the WlncR.

We do not provide timings in this section, because MATLAB-based computations are usually far from being highly optimized. Some timings will appear in the next sections for the Fortran implementation of discussed algorithms.

This example was introduced to us by Prof. Lars Eldén. In [31] more experiments with the approximation of sparse tensors by different algorithms are provided and results are compared for the truncated HOSVD, the minimal Krylov recursion, several modifications of the MKR and the WlncR Alg. 8 that was implemented by authors of [31] with minor modifications.

5.2   Compression from canonical to Tucker format

Multidimensional data often appear in modern modelling programs in canonical form (C). For example, in chemical modelling programs, e.g. PC GAMESS and MOLPRO, the *electron density function* is given as a sum of the tensor product of one-dimensional Gaussians. However, even for simple molecules, the number of terms in the decomposition obtained by MOLPRO may be too large for practically feasible computations. In order to make computations efficient, the further approximation (recompression) to the Tucker format can be performed. The accuracy of the desired approximation can vary in different applications. For some quantum chemistry problems the very precise approximation (with ten or more significant digits) is required.

Such recompression was done in [4] using the Tucker-ALS algorithm [22, 7], in [20] by the Tucker-ALS with the initial guess obtained from the coarser grids, in [11] by the Cross3D algorithm [25], in [26] by the individual cross approximation of canonical factors and in [32] by the cross approximation of Gram matrices of unfoldings. For an $n \times n \times n$ tensor given in canonical form with R terms, each tenvec costs $3nR$ operations, and the proposed versions of Alg. 7 can be applied to compute the Tucker approximation efficiently, even for large n and R.

We apply the discussed algorithm for the Tucker approximation of the electron density of some simple molecules, discretized on the uniform $n \times n \times n$ tensor grid with $n = 5121$. The convergence of algorithms, i.e. the accuracy of rank-(r, r, r) approximation for different r is shown on Figs. 2, 3. On each graph we compare the accuracy of the approximation computed by the Tucker-ALS [22, 7] (thin solid line) with the accuracy of the certain approximation method. For each method, the internal estimate of the error err is shown by thin dashed line and real accuracy $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F$ of the algorithm is shown by the thick dashed line. The core of the Tucker approximation was computed by (1). Since the evaluation of the whole $n \times n \times n$ array for $n = 5121$ requires one terabyte of memory and a lot of computational resources, we verify the accuracy of

Figure 2. Approximation accuracy of the methane electron density
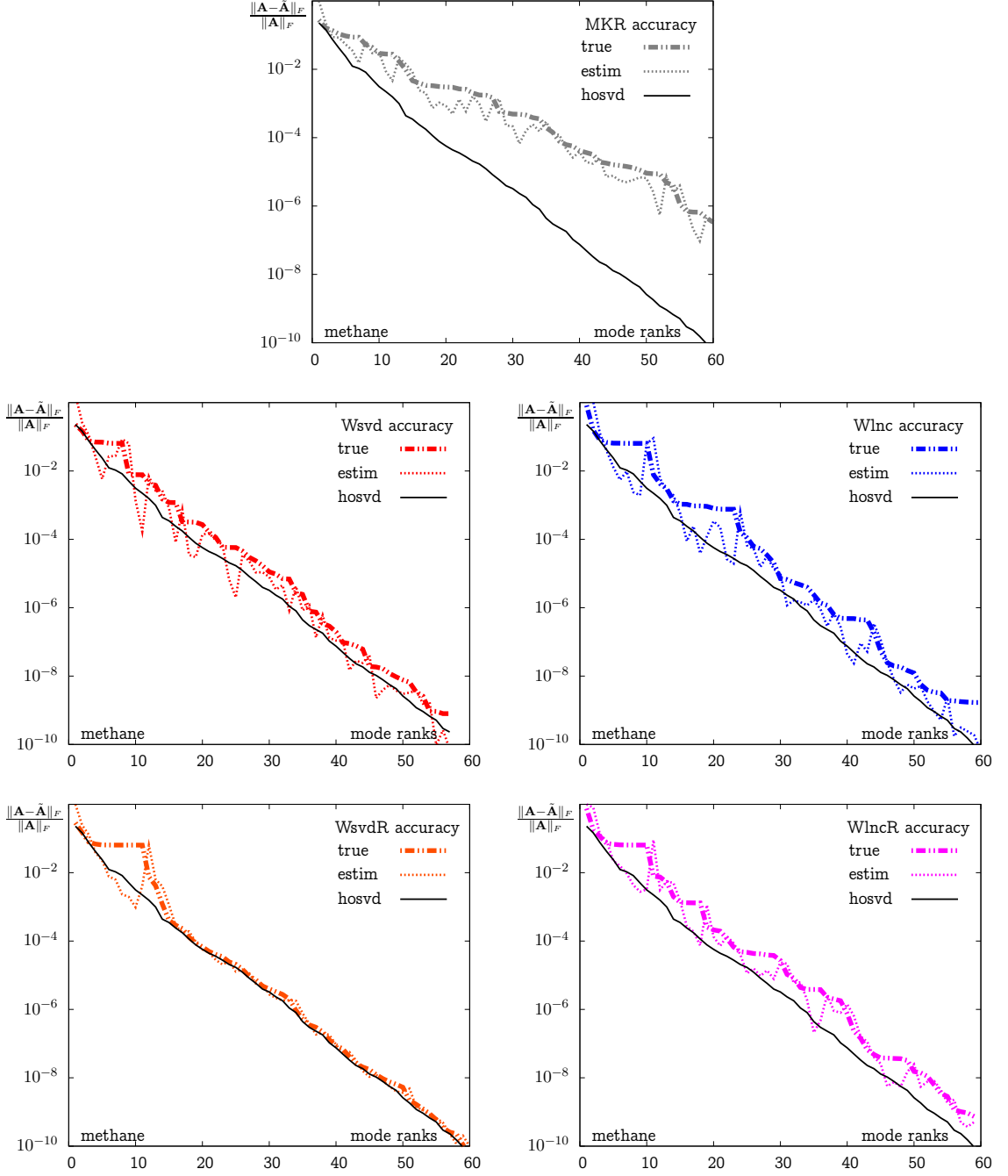
For $5121 \times 5121 \times 5121$ tensor given in the canonical form with rank 1334, we compute the Tucker approximation by algorithms, listed in Tab. 1, and plot the relative accuracy in the Frobenius norm for different values of mode ranks. On each graph the thick dashed line shows the true error of approximation, the thin dashed line shows the estimate of error and the thin solid line shows the accuracy of the HOSVD approximation for the reference.

23

Figure 3. Approximation accuracy of the glycine electron density
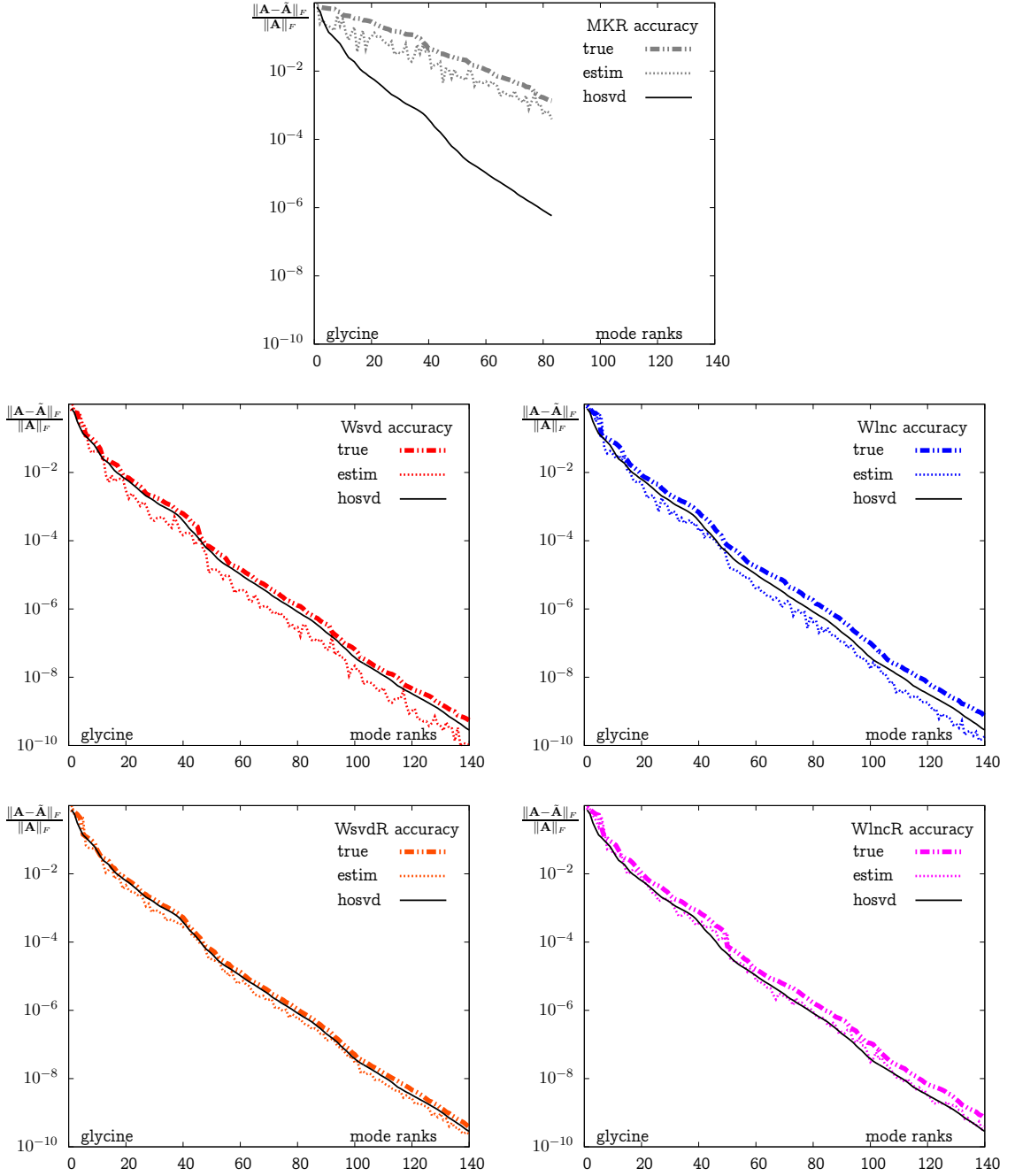
For $5121 \times 5121 \times 5121$ tensor given in the canonical form with rank 9208, we compute the Tucker approximation by algorithms, listed in Tab. 1, and plot the relative accuracy in the Frobenius norm for different values of mode ranks. On each graph the thick dashed line shows the true error of approximation, the thin dashed line shows the estimate of error and the thin solid line shows the accuracy of the HOSVD approximation for the reference.

24

Table 2. Time for approximation of electron density

molecule	accuracy	MKR	Wsvd	Wlnc	WsvdR	WlncR	cross	TALS(1)
methane	10^{-4}	1.0	6.1	4.4	2.0	0.5	1.0	2.6
R = 1334	10^{-6}	1.7	10.7	8.2	3.7	0.8	1.4	9.6
	10^{-8}	—	14.5	11.1	5.1	1.4	1.9	30
	10^{-10}	—	20.4	16.1	6.9	2.0	2.9	59
ethane	10^{-4}	2.7	17	15	6.7	1.6	3.0	4.6
R = 3744	10^{-6}	5.2	32	25	12	2.8	4.9	17
	10^{-8}	—	45	35	17	3.9	6.3	42
	10^{-10}	—	61	50	23	5.3	8.2	83
ethanol	10^{-4}	8.0	54	45	17	4.7	8.1	22
R = 6945	10^{-6}	14	89	74	30	8.4	13	81
	10^{-8}	—	135	108	45	13	17	194
	10^{-10}	—	180	145	61	18	22	391
glycine	10^{-4}	—	85	69	37	7.5	24	32
R = 9208	10^{-6}	—	131	112	57	13	33	96
	10^{-8}	—	200	160	80	18	43	211
	10^{-10}	—	268	211	114	24	60	412

For $5121 \times 5121 \times 5121$ tensor given in the canonical form with rank R, we compute the Tucker approximation with different relative accuracy bound by different algorithms and show time in seconds. Algorithms MKR, Wsvd, Wlnc, WsvdR, WlncR are listed in Tab. 1, 'cross' algorithm is based on individual cross approximation of canonical factors [26], 'TALS(1)' is time for one iteration of the Tucker-ALS method [22, 7]

algorithms by comparison of the result with the Tucker approximation computed by the individual cross approximation of canonical factors [26] with the accuracy set to $\varepsilon = 10^{-12}$. The verification of the latter was done in [26] by the exhaustive verification on cluster platforms. The residual between two tensors in the Tucker format is computed as proposed in [27].

We note the slow convergence of the MKR for the methane electron density and the breakdown for glycine. All versions of the Wedderburn elimination converge much better, and for the larger glycine molecule the convergence is even more regular, than for methane. Accuracy of methods with the Lanczos-like pivoting is close to the optimal one, and accuracy of method with the restricted SVD-like pivoting is almost equal to optimal, except for first steps of the process, when accumulated subspaces are small and impose significant restrictions on the selection of leading vectors. Also, the internal error value, estimated by norm of $\mathbf{A} \times_1 x_k^\mathsf{T}$ (and the same for other modes), is less regular than the real accuracy, that decays monotonically. In the methods with unrestricted pivoting (see Wlnc) the internal error value is computed by the spectral norm of matrix $\mathbf{A} \times_1 x_k^\mathsf{T}$ and appears to be 'more optimistic' than the real error, that is measured in

the Frobenius norm. In methods with the restricted pivoting the Frobenius norm of $\mathbf{A} \times_1 x_{k+1}^{\mathsf{T}} \times_2 Y_l^{\mathsf{T}} \times_3 Z_m^{\mathsf{T}}$ is used to estimate the internal error and matches the real error more closely.

Timings for the approximation for different molecules and accuracy parameters are given in Tab. 2. We compare all methods listed in Tab. 1 and the method proposed in [26] based on incomplete cross approximation [36] of unfoldings. For Wedderburn elimination methods we set $p_{als} = p_{pow} = 3$. From the preliminary experiments we see that this rather small number of 'inner' iterations is sufficient; the experiments with p_{als} and p_{pow} set to 10, 30 and 300 show almost the same convergence and final accuracy for all the molecules. For the reference we also provide time for one iteration of the Tucker-ALS method [22, 7], that uses output of Alg. 8 as an initial guess. In our implementation of the Tucker-ALS the low-rank decomposition of (19) is computed by the cross approximation, that is usually several $(2 \div 20)$ times faster than the SVD-based computations. However, even then one iteration of the Tucker-ALS seems to be quite expensive. In practical computations several (usually $3 \div 20$) iterations of the Tucker-ALS are required depending on the accuracy of the initial guess; it is also necessary to estimate the ranks of the desired approximation before starting the Tucker-ALS. In section 5.1 we explain that each Tucker-ALS iteration requires $3r^2$ tenvecs, while Wedderburn methods require $\mathcal{O}(r)$ tenvecs to compute Tucker factors U, V, W and r^2 tenvecs to generate the core $\mathbf{G}$ (see Tab. 1). Therefore, for large ranks and fixed p_{pow} and p_{als}, Wedderburn methods become more efficient in comparison with a single iteration of the Tucker-ALS. We can note this behaviour in Tab. 2 on the lines corresponding to the high precision of approximation, where the ranks of the approximation are also high.

We note that for this example the Wedderburn elimination with the restricted Lanczos-like pivoting (Alg. 8) is faster than all other versions of the Wedderburn elimination method. We also note that it outperforms the minimal Krylov recursion, since Alg. 1 converges slowly (or even does not converge) and uses more iterations to reach the same accuracy level. WlncR also outperforms method based on cross approximation of unfoldings [26]. We conclude that for this problem Alg. 8 outperforms previously proposed methods.

5.3   Recompression in operations with structured tensors

The efficient operations with matrices and vectors in compressed tensor formats is crucial in the construction of efficient iterative methods for solving equations and eigenproblems in three and more dimensions. The approach to such highly-efficient tensor linear algebra subroutines was discussed in [27, 33], where it is shown that the efficient evaluation of all basic linear algebra subroutines with tensor-structured data is based on the fast recompression of certain structured tensor. As an example, consider Hadamard (elementwise) multiplication between $n_1 \times n_2 \times n_3$ tensors

$$\mathbf{A} = \mathbf{G} \times_1 U^{(A)} \times_2 V^{(A)} \times_3 W^{(A)}, \qquad \mathbf{B} = \mathbf{H} \times_1 U^{(B)} \times_2 V^{(B)} \times_3 W^{(B)}.$$

Let mode ranks of $\mathbf{A}$ be r_1, r_2, r_3 and mode ranks of $\mathbf{B}$ be p_1, p_2, p_3. The result reads

$$\mathbf{C} = \mathbf{F} \times_1 U \times_2 V \times_3 W \qquad (20)$$

with $p_1 r_1 \times p_2 r_2 \times p_3 r_3$ core $\mathbf{F} \overset{\text{def}}{=} \mathbf{Kron}(\mathbf{G}, \mathbf{H})$ and *non-orthogonal* factors U, V, W of sizes $n_1 \times p_1 r_1$, $n_2 \times p_2 r_2$ and $n_3 \times p_3 r_3$, respectively. Formally this is again the Tucker format with the core and factors given by

$$\mathbf{F}(ap, bq, cs) \overset{\text{def}}{=} \mathbf{G}(p, q, s)\mathbf{H}(a, b, c), \qquad U(i, ap) = U^{(A)}(i, p)U^{(B)}(i, a),$$

and so on for V, W. The mode ranks of $\mathbf{C}$ are products of correspondent mode ranks of $\mathbf{A}$ and $\mathbf{B}$, and recompression is required to reduce the storage size.

In [33] the fast recompression method based on the individual filtering of factors was proposed. Numerical examples in [33] include evaluation of the Hartree potential for the electron density of molecules, discussed in section 5.2. This problem writes as a multiplication between three-level matrix given in the canonical format (with diagonal core tensor) and three-dimensional vector of the electron density given in the Tucker format. The computation requires $10 \div 120$ seconds depending on the complexity of molecule and desired accuracy of evaluation. However, examples of Tucker to Tucker multiplication were not presented in [33], since this operation appears to be sufficiently more expensive. For the large molecules it requires up to an hour, that we consider not affordable.

We show that fast and accurate multiplication between tensors given in the Tucker format can be done using Wedderburn-based methods. Each tenvec operation with tensor (20) can be done in $\mathcal{O}(q^4 + nq^2)$, where $q = \max(p, r)$, that is fast enough even for n up to hundred thousands and p, r up to several hundreds. We apply the discussed algorithm for the Hadamard multiplication of the discretized electron density of simple molecules to themselves. This operation can be a building block for algorithms that compute pointwise nonlinear functions of large tensors with linear in mode size complexity. One of the important applications is the cubic root of the electron density that appear in the Kohn-Sham model. A good initial guess for such methods can be evaluated by the mimic algorithm [26].

The convergence of algorithms, i.e. the accuracy of rank-(r, r, r) approximation for different r is shown on Figs. 4, 5. On each graph we compare accuracy of the approximation computed by the Tucker-ALS [22, 7] (thin solid line) with the accuracy of certain approximation method. The real accuracy $\|\mathbf{A} - \tilde{\mathbf{A}}\|_F$ was verified by comparing the result with the Tucker approximation computed by the Cross3D algorithm [25] with the accuracy set to $\varepsilon = 10^{-12}$. The Cross3D algorithm was verified in [25, 11] by exhaustive check on cluster platforms.The residual between two Tucker formats is computed as proposed in [27].

As well as for recompression of electron density from canonical form, in this problem all versions of the Wedderburn elimination demonstrate good convergence, and for larger glycine molecule it is even more regular, than for methane. Comparing the different versions of Alg. 7, we note the same behaviour that is already described in section 5.2.

Figure 4. Approximation accuracy of the Hadamard square of the methane electron density
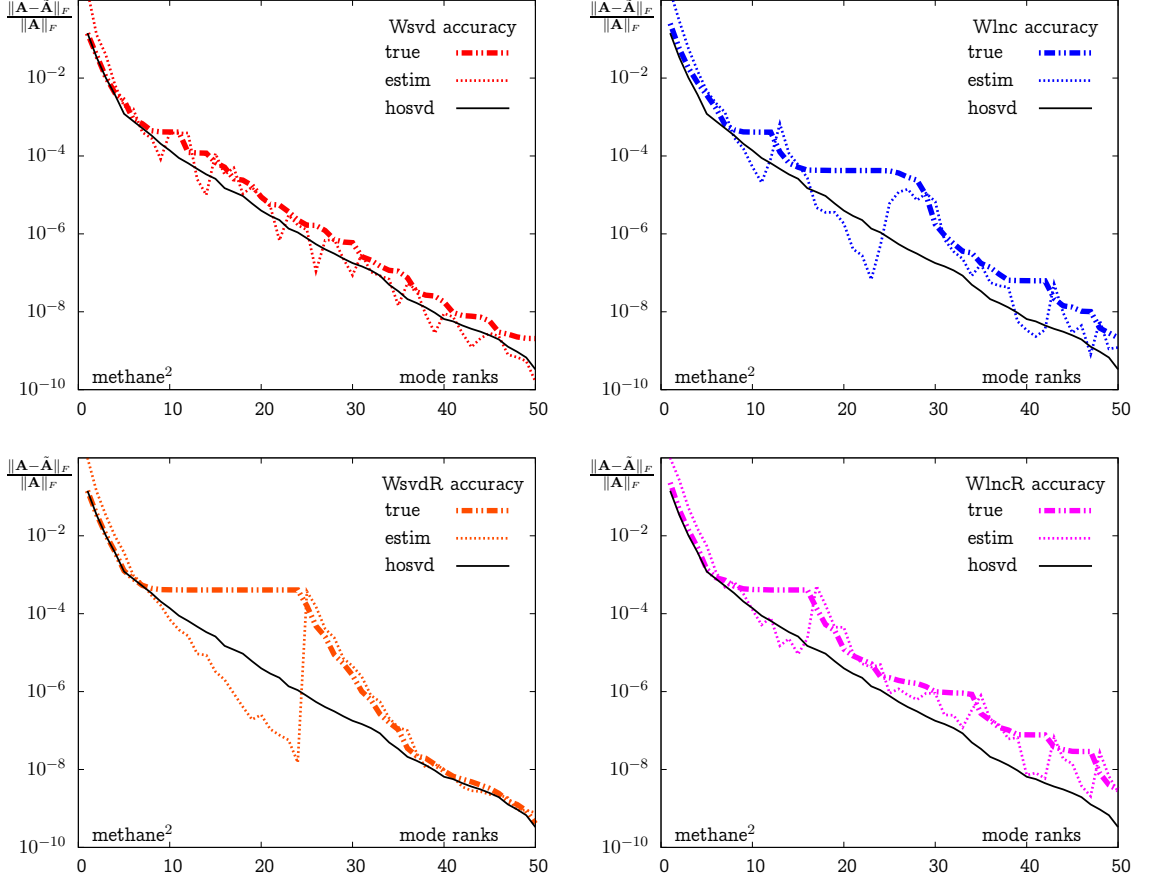
For the Hadamard square of the $5121 \times 5121 \times 5121$ tensor given in the Tucker format with mode ranks $(74, 74, 74)$, we compute Tucker approximation by algorithms, listed in Tab. 1, and plot the relative accuracy in the Frobenius norm for different values of mode ranks. On each graph the thick dashed line shows the true error of approximation, the thin dashed line shows the estimate of error and the thin solid line shows the accuracy of the HOSVD approximation for the reference.

Figure 5. Approximation accuracy of the Hadamard square of the methane electron density
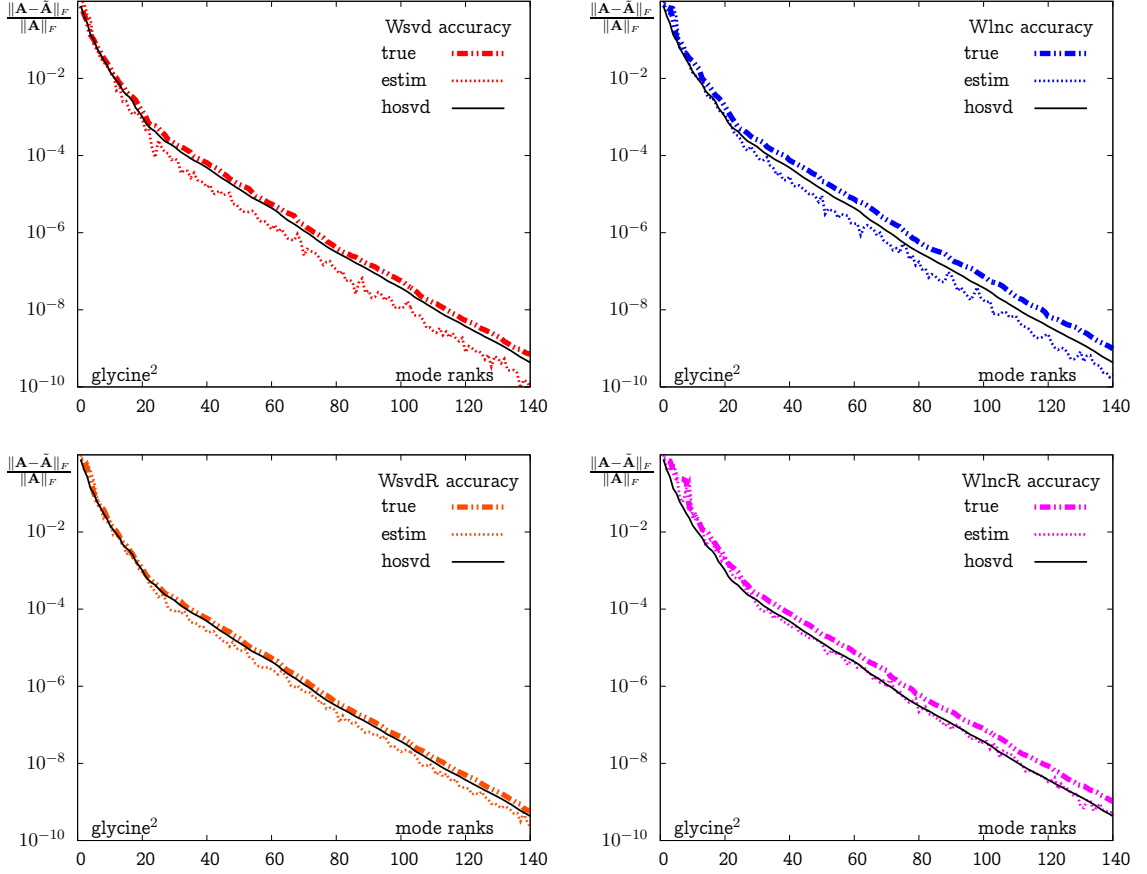
For the Hadamard square of the $5121 \times 5121 \times 5121$ tensor given in the Tucker format with mode ranks $(62, 176, 186)$, we compute Tucker approximation by algorithms, listed in Tab. 1, and plot the relative accuracy in the Frobenius norm for different values of mode ranks. On each graph the thick dashed line shows the true error of approximation, the thin dashed line shows the estimate of error and the thin solid line shows the accuracy of the HOSVD approximation for the reference.

Table 3. Time for approximation of electron density

molecule	accuracy	Wsvd	Wlnc	WsvdR	WlncR	TALS(1)
methane	10^{-4}	15.5	13.8	8.1	2.8	2.4
$(74, 74, 74)$	10^{-6}	34	33	14.8	9.7	14.3
	10^{-8}	63	46	43	18.6	42
	10^{-10}	93	74	68	40	97
ethane	10^{-4}	22	20	11.8	4.2	4.2
$(67, 94, 83)$	10^{-6}	46	41	33	14.0	16.7
	10^{-8}	82	68	72	27	45
	10^{-10}	125	105	127	56	117
ethanol	10^{-4}	120	101	106	45	52
$(128, 127, 134)$	10^{-6}	281	228	293	176	257
	10^{-8}	493	419	635	441	678
	10^{-10}	736	653	1100	808	1370
glycine	10^{-4}	179	170	177	60	64
$(62, 176, 186)$	10^{-6}	442	380	600	217	270
	10^{-8}	732	600	1033	500	646
	10^{-10}	1010	850	1530	888	1223

For the Hadamard square of the $5121 \times 5121 \times 5121$ tensor given in Tucker form, we compute Tucker approximation with different relative accuracy bound by different algorithms and show time in seconds. Algorithms Wsvd, Wlnc, WsvdR, WlncR are listed in Tab. 1, 'TALS(1)' is time for one iteration of Tucker-ALS method [22, 7]

Timings for the approximate computation of the Tucker-to-Tucker multiplication for different methods and accuracy parameters are given in Tab. 3. We compare all methods listed in Tab. 1 and provide time for one iteration of the Tucker-ALS method [22, 7], that uses output of Alg. 8 as an initial guess. For the Wedderburn elimination methods we set $p_{als} = p_{pow} = 3$. Again, we note that for this example the WlncR Alg. 8 is faster than all other versions of Wedderburn elimination algorithms and also than one iteration of Tucker-ALS method. In this case the timings of different versions of Wedderburn elimination algorithms are more similar, because the cost of the evaluation of the core (r^2 tenvecs) dominates over $\mathcal{O}(nr^2)$ time for additional operations. Nevertheless, we conclude that for this problem Alg. 8 can be method of choice for fast approximate evaluation of operations with data in tensor formats.

6   Conclusion and further work

We presented the family of algorithms for the approximation of large three-dimensional tensor that access it only using the tenvec operation (tensor-by-vector-by-vector mul-

tiplication). Our approach is based on the Wedderburn rank-reduction formula and admits different strategies to select vectors of the Wedderburn elimination ('pivoting') that lead to algorithms with rather different convergence and complexity estimates. The fastest algorithm from presented family, namely WlncR Alg. 8, may converge slowly or stagnate on the first steps of iterations. However, one can propose an efficient algorithm combining more efficient but demanding pivoting strategy on the first steps (for example, Wsvd, that is free from breakdowns) with the fast WlncR pivoting strategy on the next steps of algorithm.

The presented methods can be applied for the approximation of dominant subspaces of large structured tensors. In the provided numerical examples some of the proposed algorithms are much faster than the Tucker-ALS [22, 7] algorithm and as fast as the minimal Krylov recursion [30], but more accurate in certain cases. The proposed methods can be directly applied for the fast computation of bilinear operations between structured tensors, but the efficiency can be further improved by combining them with the individual factor filtering proposed in [33].

Canonical and Tucker formats can be straightforwardly generalized to d dimensions, however each of them has serious drawbacks, and another decomposition should be used for high dimensions, for example recently introduced tensor-train (TT decomposition, see [28, 23, 24]), that are based on the SVD techniques, but is free from the curse of dimensionality. Therefore, it is very natural to extend the proposed ideas to the TT format. In the sequel we will show how to construct algorithm for TT format that use tensor through tensor-by-vectors multiplication, i.e. Krylov-type methods in d dimensions.

It is also important to analyse how does the choice of parameters p_{pow} and p_{als} change the convergence properties and final accuracy of the proposed methods. The preliminary experiments show that for the experiments provided in this paper it is sufficient to take $p_{als} = p_{pow} = 3$. The deeper theoretical analysis is required, that should generalize the theory of Arnoldi methods for the tensor case. This is a topic of a forthcoming work.

Acknowledgements

References

[1] B. W. BADER AND T. G. KOLDA, *Efficient MATLAB computations with sparse and factored tensors*, SIAM J. Sci. Comp., 30 (2007), pp. 205–231.

[2] A. BOUARICHA, *Tensor-Krylov methods for large nonlinear equations*, Comp. Optimization Appl., 5 (1996), pp. 207–232.

[3] J. D. CAROLL AND J. J. CHANG, *Analysis of individual differences in multi-dimensional scaling via n-way generalization of Eckart-Young decomposition*, Psychometrika, 35 (1970), pp. 283–319.

[4] S. R. CHINNAMSETTY, H.-J. FLAD, V. KHOROMSKAIA, AND B. N. KHOROMSKIJ, *Tensor decomposition in electronic structure calculations on 3D Cartesian grids*, J. Comp. Phys., 228 (2009), pp. 5749–5762.

[5] M. T. CHU, R. E. FUNDERLIC, AND G. H. GOLUB, *A rank-one reduction formula and its applications to matrix factorizations*, SIAM Review, 37 (1995), pp. 512–530.

[6] L. DE LATHAUWER, B. DE MOOR, AND J. VANDEWALLE, *A multilinear singular value decomposition*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1253–1278.

[7] ——, *On best rank-1 and rank-($R_1, R_2, ..., R_N$) approximation of high-order tensors*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1324–1342.

[8] V. DE SILVA AND L. LIM, *Tensor rank and the ill-posedness of the best low-rank approximation problem*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 1084–1127.

[9] A. DEFANT AND K. FLORET, *Tensor norms and operator ideals*, North Holland, 1993.

[10] L. ELDÉN AND B. SAVAS, *A Newton-Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 248–271.

[11] H.-J. FLAD, B. N. KHOROMSKIJ, D. V. SAVOSTYANOV, AND E. E. TYRTYSHNIKOV, *Verification of the cross 3D algorithm on quantum chemistry data*, Rus. J. Numer. Anal. Math. Model., 23 (2008), pp. 329–344.

[12] G. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[13] G. GOLUB AND C. VAN LOAN, *Matrix computations*, Johns Hopkins University Press, 1996.

[14] S. Goreinov, I. Oseledets, D. Savostyanov, E. Tyrtyshnikov, and N. Zamarashkin, *How to find a good submatrix*, in Matrix Methods: Theory, Algorithms, Applications, V. Olshevsky and E. Tyrtyshnikov, eds., World Scientific Publishing, 2010, pp. 247–256.

[15] S. A. Goreinov and E. E. Tyrtyshnikov, *The maximal-volume concept in approximation by low-rank matrices*, Contemporary Mathematics, 208 (2001), pp. 47–51.

[16] A. Greenbaum, *Iterative methods for solving linear systems*, Society for Industrial Mathematics, 1997.

[17] R. A. Harshman, *Foundations of the Parafac procedure: models and conditions for an explanatory multimodal factor analysis*, UCLA Working Papers in Phonetics, 16 (1970), pp. 1–84.

[18] F. L. Hitchcock, *The expression of a tensor or a polyadic as a sum of products*, J. Math. Phys, 6 (1927), pp. 164–189.

[19] M. Ishteva, L. De Lathauwer, P. A. Absil, and S. Van Huffel, *Differential-geometric newton method for the best rank-(r_1, r_2, r_3) approximation of tensors*, Numerical Algorithms, 51 (2009), pp. 179–194.

[20] B. N. Khoromskij and V. Khoromskaia, *Multigrid accelerated tensor approximation of function related multidimensional arrays*, SIAM J. Sci. Comp., 31 (2009), pp. 3002–3026.

[21] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[22] P. Kroonenberg and J. de Leeuw, *Principal component analysis of three-mode data by means of alternating least squares algorithms*, Psychometrika, 45 (1980), pp. 69–97.

[23] I. V. Oseledets, *Compact matrix form of the d-dimensional tensor decomposition*, SIAM J. Sci. Comp, 31 (2009), pp. 3744–3759.

[24] ——, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145.

[25] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov, *Tucker dimensionality reduction of three-dimensional arrays in linear time*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 939–956.

[26] I. V. Oseledets, D. V. Savostyanov, and E. E. Tyrtyshnikov, *Cross approximation in tensor electron density computations*, Numer. Lin. Alg. Appl., (2009).

[27] ——, *Linear algebra for tensor problems*, Computing, 85 (2009), pp. 169–188.

[28] I. V. OSELEDETS AND E. E. TYRTYSHNIKOV, *Breaking the curse of dimensionality, or how to use svd in many dimensions*, SIAM J. Sci. Comp., 31 (2009), pp. 3744–3759.

[29] C. C. PAIGE, *Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix*, IMA J. Appl. Math., 18 (1976), p. 341.

[30] B. SAVAS AND L. ELDÉN, *Krylov subspace methods for tensor computations*, Preprint LITH-MAT-R-2009-02-SE, Dep. Math. Linköpings Univ., Linköping, 2009.

[31] ——, *Krylov subspace methods for tensor computations*, Preprint 1005.0683v2 [math.NA], 6 May 2010.

[32] D. V. SAVOSTYANOV, *Fast revealing of mode ranks of tensor in canonical format*, Numer. Math. Theor. Meth. Appl., 2 (2009), pp. 439–444.

[33] D. V. SAVOSTYANOV AND E. E. TYRTYSHNIKOV, *Approximate multiplication of tensor matrices based on the individual filtering of factors*, J. Comp. Math. Math. Phys., 49 (2009), pp. 1662–1677.

[34] A. L. TRAUD, E. D. KELSIC, P. J. MUCHA, AND M. A. PORTER, *Community structure in online collegiate social networks*, (2008), arXiv:physics.soc-ph/0809.0690.

[35] L. R. TUCKER, *Some mathematical notes on three-mode factor analysis*, Psychometrika, 31 (1966), pp. 279–311.

[36] E. E. TYRTYSHNIKOV, *Incomplete cross approximation in the mosaic–skeleton method*, Computing, 64 (2000), pp. 367–380.

[37] J. H. M. WEDDERBURN, *Lectures on matrices, colloquim publications*, vol. XVII, AMS, New York, 1934.

[38] T. ZHANG AND G. H. GOLUB, *Rank-one approximation to high-order tensors*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 534–550.